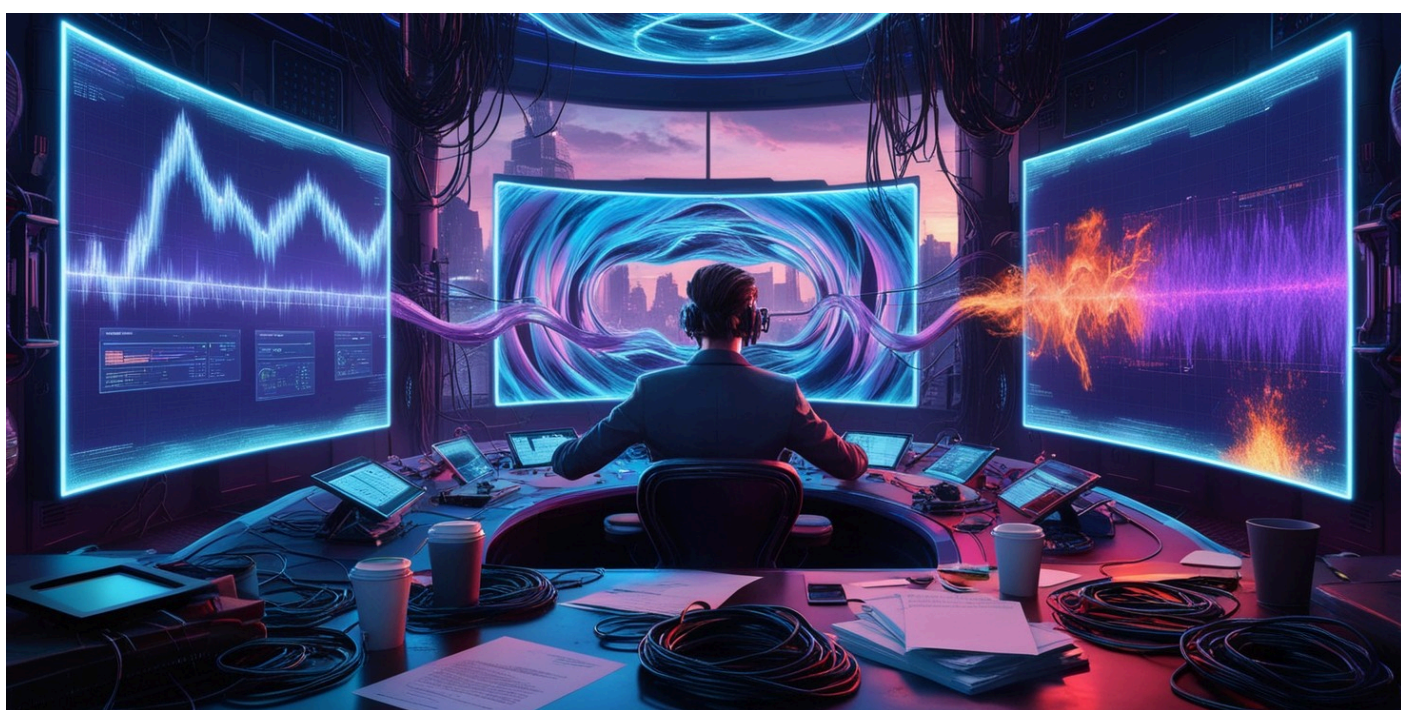


Прогнозирование временных рядов в Python



Введение в анализ временных рядов

Анализ временных рядов (Time series analysis, TSA) – это метод изучения характеристик целевой переменной во времени, где **время** является **независимой переменной**.

Другими словами, TSA позволяет нам **прогнозировать будущие значения** целевой переменной на основе исторических данных.

TSA используется в различных областях, например, для прогнозирования погодных показателей (температуры, осадков, давления и т.д.), финансовых рынков (цен на акции, курсов валют, процентных ставок и т.д.), продаж (спроса, объемов производства, запасов и т.д.).

Этапы анализа и прогнозирования временных рядов

Признаки временных рядов

Шаг временного ряда

Лag временного ряда

Компоненты и ограничения анализа временных рядов

Компоненты временных рядов

Ограничения анализа временных рядов

Методы проверки стационарности временных рядов

Тест Дикки-Фуллера с поправкой на автокорреляцию (ADF-тест)

Тест Квитка-Филлипса-Шмидта-Шина (KPSS-тест)

Преобразование нестационарных данных в стационарные

Алгоритмы анализа временных рядов

Авторегрессивная модель (Auto-Regressive Model, AR)

Модель скользящего среднего (Moving Average)

Модели ARMA и ARIMA

Этапы анализа и прогнозирования временных рядов

1. Подготовка данных: сбор, загрузка, предобработка (работа с пропусками, дубликатами, неправильными типами и т.д.) данных.
2. Построение визуализаций для выявления трендов, сезонности и других закономерностей.
3. Анализ стационарности.
4. Выбор и разработка модели прогнозирования.
5. Прогнозирование будущих значений целевой переменной.
6. Оценка качества модели и её корректировка при необходимости.

Для иллюстрации методов анализа и прогнозирования временных рядов обратимся к данным из соревнования Kaggle [Sore Sales - Time Series Forecasting](#), содержащим информацию о продажах товаров. В частности, мы будем использовать два файла из этого датасета: oil.csv и train.csv.

Структура данных:

1. oil.csv: файл содержит информацию о ценах на нефть в различные дни.
 - date : Дата наблюдения.
 - dcoilwtico : цена нефти на эту дату.

2. train.csv: файл содержит информацию о продажах товаров в магазинах.

- id : уникальный идентификатор наблюдения.
- date : дата продажи.
- store_nbr : номер магазина.
- family : категория товара.
- sales : количество проданных единиц товара.
- onpromotion : признак акции (1 - товар находится на акции, 0 - нет).

Начнем с загрузки и предобработки данных.

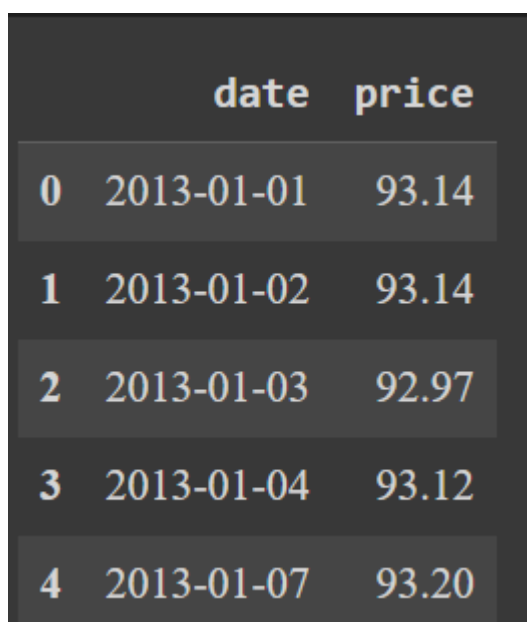
```
# загружаем ZIP-архив с данными с GitHub по указанному URL-адресу
# и затем распаковываем его в указанную папку (/content/)
!wget https://github.com/dkalenov/store-sales-time-series-forecasting/raw/main/store-sales-time-series-forecasting.zip
!unzip store-sales-time-series-forecasting.zip -d /content/

import pandas as pd

# Считываем данные, переименовываем столбец и заполняем пропуски
oil_data = pd.read_csv('oil.csv').rename({'dcoilwtico': 'price'}, axis = 1).
df = oil_data.copy()

# Преобразуем столбец 'date' в формат datetime
df['date'] = pd.to_datetime(df['date'])

df.head(3)
```



	date	price
0	2013-01-01	93.14
1	2013-01-02	93.14
2	2013-01-03	92.97

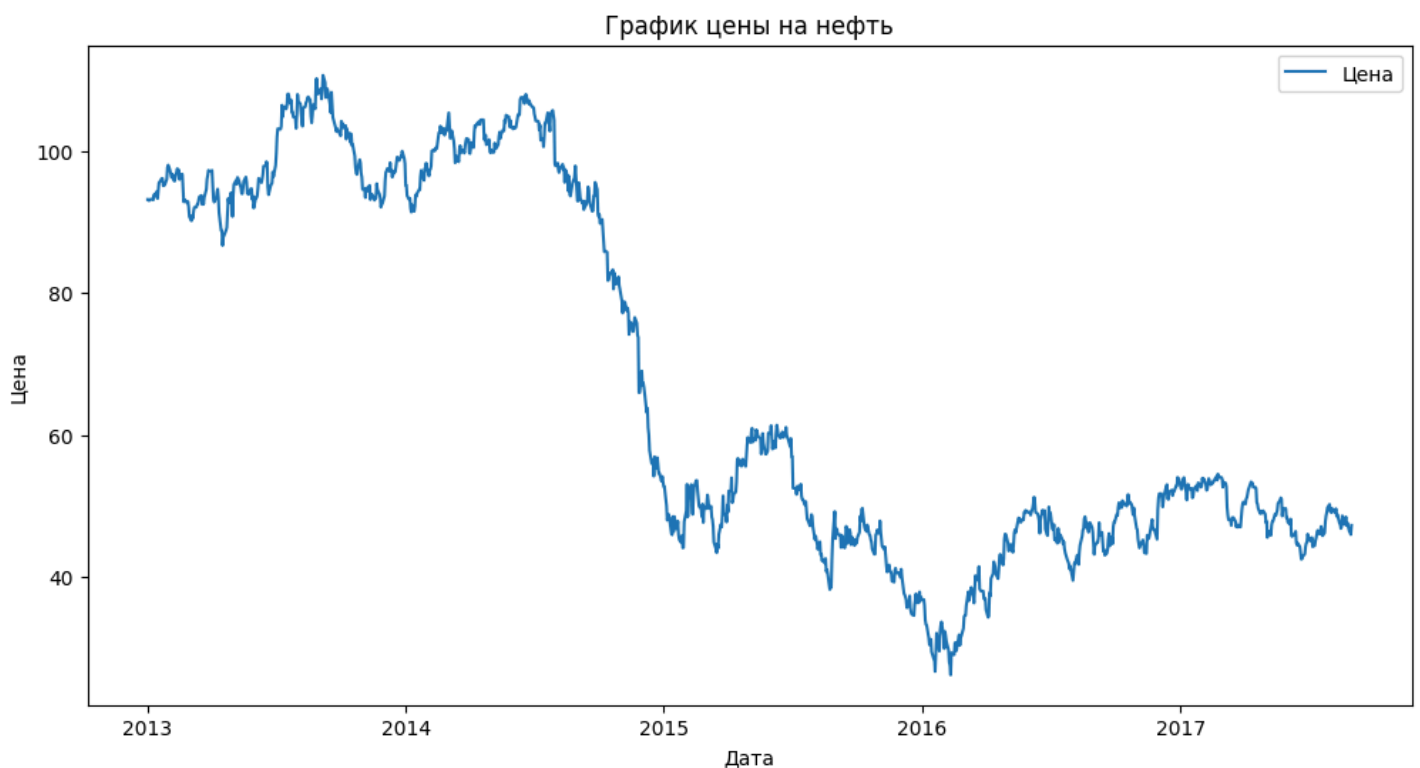
Визуализируем динамику цен с течением времени:

```
import matplotlib.pyplot as plt

# Задаём размер графика
plt.figure(figsize=(12, 6))

# Построение линейчатого графика временного ряда
plt.plot(df['date'], df['price'], label='Цена')

# Устанавливаем заголовки и легенду
plt.title('График цены на нефть')
plt.xlabel('Дата')
plt.ylabel('Цена')
plt.legend()
plt.show()
```



Для дальнейшей работы с временными рядами рассмотрим два вида уникальных признака характерных для временных рядов: временной шаг и лаг.

Признаки временных рядов

Шаг временного ряда

Шаг временного ряда — это признак, которые можно напрямую вывести из временной метки (секунды, минуты, часы, дни, недели и т.д.). Самый базовый признак временного шага — это фиктивная переменная времени (time dummy), которая отсчитывает шаги времени в ряду от начала до конца.

Добавим новый столбец `time`, который представляет собой массив чисел от 0 до длины датафрейма:

```
import numpy as np

df['time'] = np.arange(len(df.index))
df
```

	date	price	time
0	2013-01-01	93.14	0
1	2013-01-02	93.14	1
2	2013-01-03	92.97	2
3	2013-01-04	93.12	3
4	2013-01-07	93.20	4
...	...	...	...
1213	2017-08-25	47.65	1213
1214	2017-08-28	46.40	1214
1215	2017-08-29	46.46	1215
1216	2017-08-30	45.96	1216
1217	2017-08-31	47.26	1217

1218 rows × 3 columns

Визуализируем данные:

```
import matplotlib.pyplot as plt
import seaborn as sns

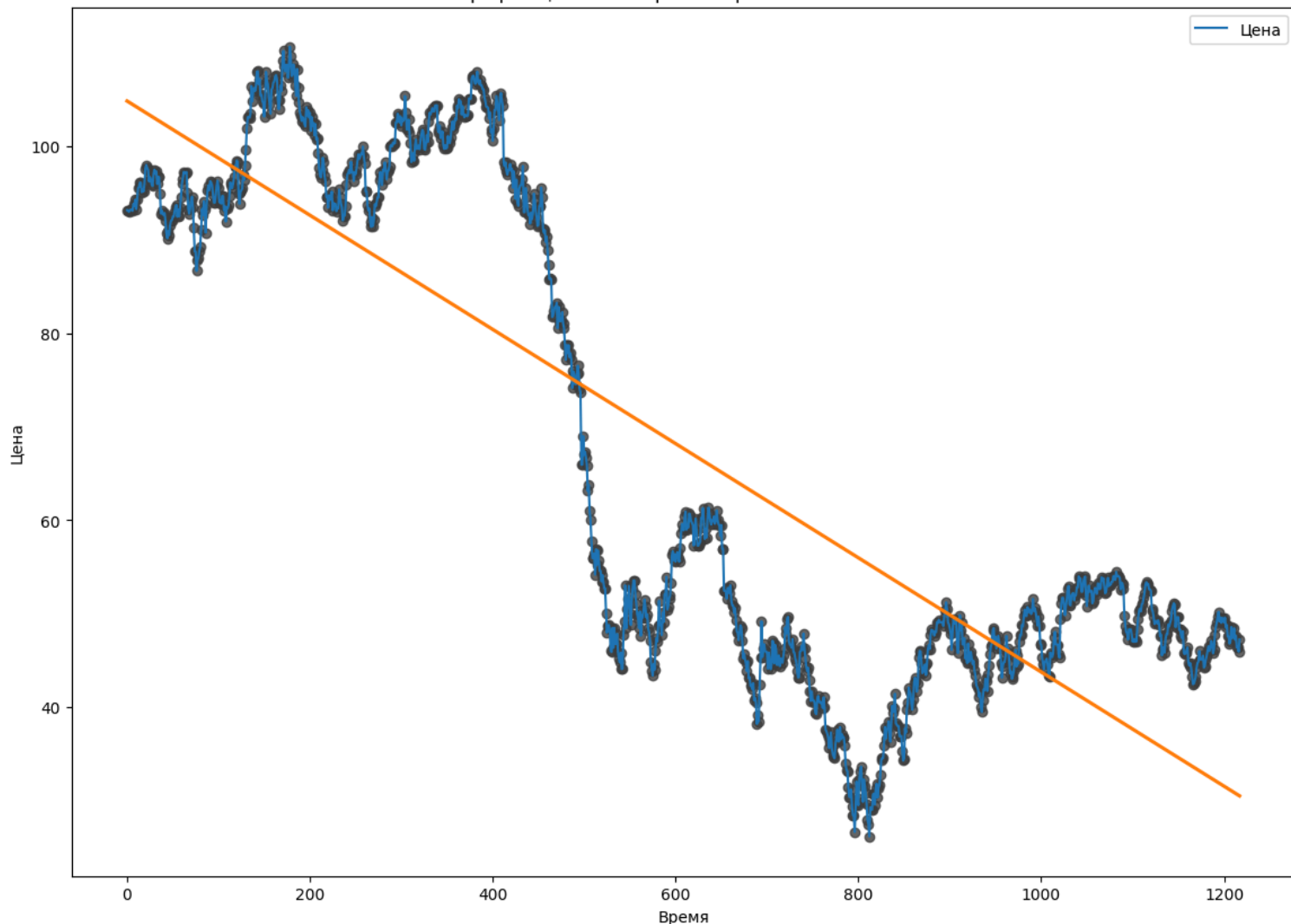
# Задаём размер визуализации
fig, ax = plt.subplots(figsize=(14, 10))

# Построение линейчатого графика временного ряда
ax.plot(df['time'], df['price'], label='Цена')

# Добавление линии регрессии с помощью Seaborn
sns.regplot(x='time', y='price', data=df, ci=None, scatter_kws=dict(color='0

ax.set_title('График цены на нефти по временным шагам')
ax.set_xlabel('Время')
ax.set_ylabel('Цена')
ax.legend()
plt.show()
```

График цены на нефти по временным шагам



Шаг временного ряда позволяют моделировать зависимость от времени.

Лег временного ряда

Для создания признака с лагом временного ряда мы смещаем наблюдения целевого признака таким образом, чтобы они выглядели как произошедшие позже во времени. В данном примере мы создадим лаг с шагом 1, хотя возможно смещение и на несколько шагов. В Python это можно реализовать с помощью метода `shift()`. Он позволяет сдвигать значения в столбце вниз (или вверх) на заданное количество периодов.

```
df['lag_1'] = df['sales'].shift(1)
df.head()
```

	date	price	time	lag_1
0	2013-01-01	93.14	0	NaN
1	2013-01-02	93.14	1	93.14
2	2013-01-03	92.97	2	93.14
3	2013-01-04	93.12	3	92.97
4	2013-01-07	93.20	4	93.12

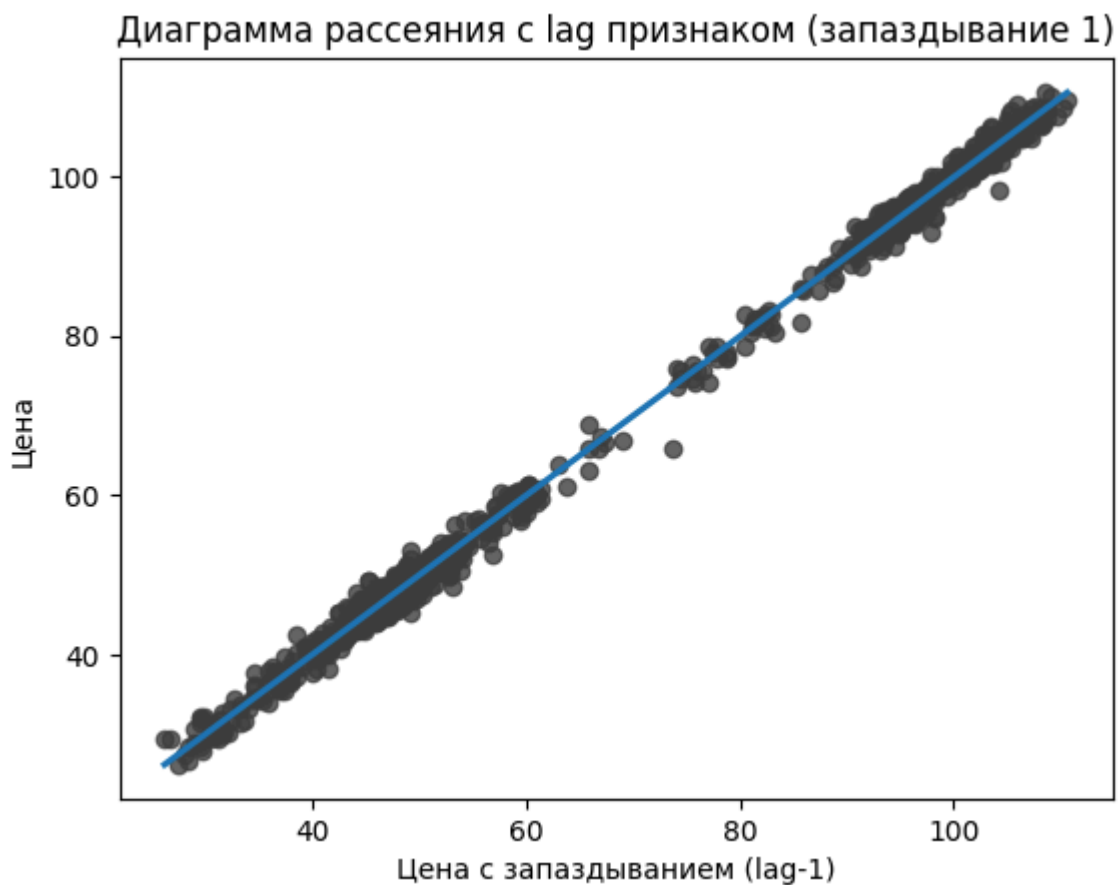
При создании признака с лагом нам необходимо решить, что делать с появившимися пропущенными значениями. Один из вариантов – заполнить пропуски (нулями, средним значением и тд.), либо просто удалить строки с пропущенными значениями. Мы удалим:

```
df.dropna(inplace=True)

fig, ax = plt.subplots()
ax = sns.regplot(x='lag_1', y='price', data=df, ci=None, scatter_kws=dict(co

# Устанавливаем заголовок и подписи осей
ax.set_title('Диаграмма рассеяния с lag признаком (запаздывание 1)')
ax.set_xlabel('Цена с запаздыванием (lag-1)')
ax.set_ylabel('Цена')

plt.show()
```

Из точечного графика видно, что цена на нефть в один день сильно коррелирует с ценой за предыдущий день. В подобных случаях использование лагов может быть крайне полезным.

В более общем смысле, лаги позволяют моделировать серийную зависимость. Временной ряд обладает серийной зависимостью, когда наблюдение можно предсказать на основе предыдущих наблюдений. В случае с ценой на нефть мы можем прогнозировать, что высокая цена в один день, как правило, означают высокую цену на следующий день.

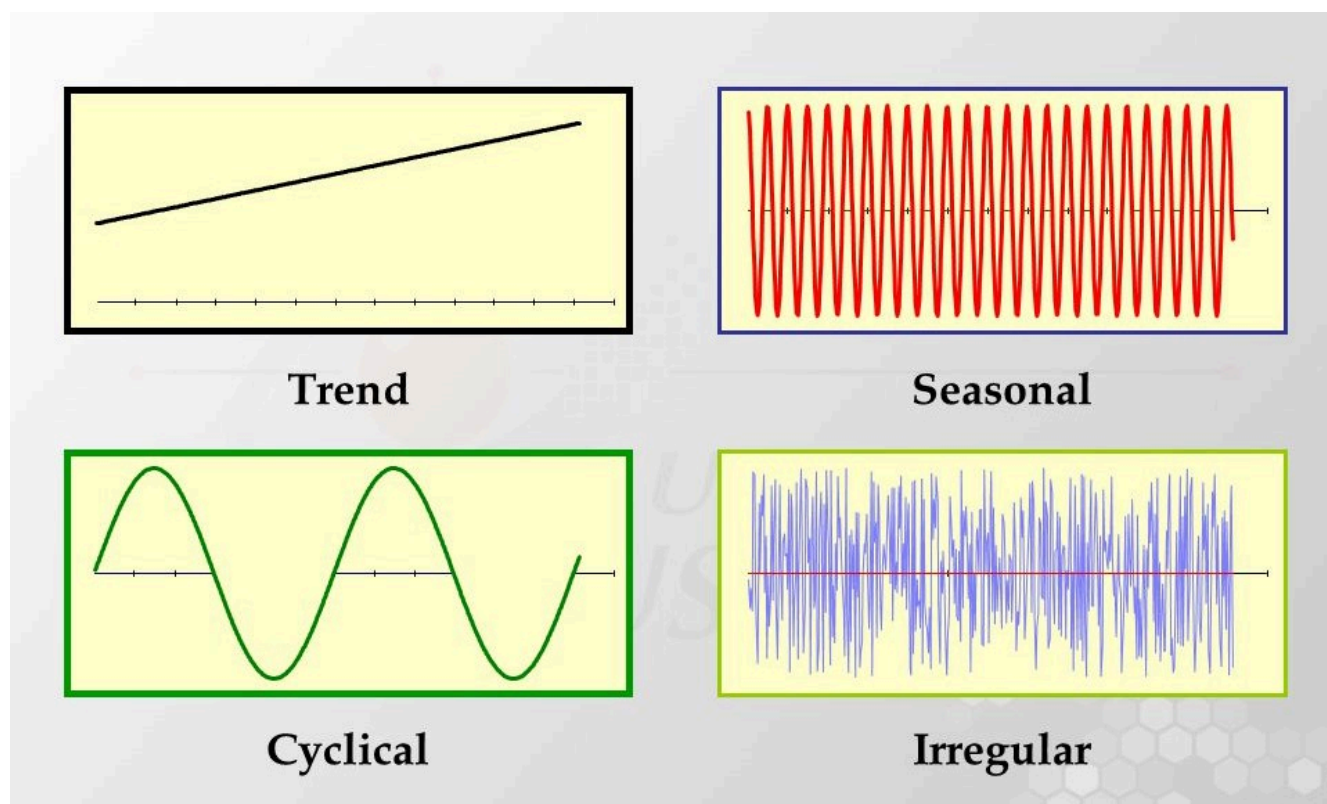
Компоненты и ограничения анализа временных рядов

Мы рассмотрели некоторые признаки временных рядов, которые можно создать для моделирования. А теперь перейдем к компонентам анализа временных рядов, которые являются более фундаментальными характеристиками данных.

Компоненты временных рядов

- Тренд (Trend): показывает общее направление движения данных временного ряда на протяжении длительного времени. Это может быть восходящий тренд (увеличение значений), нисходящий тренд (уменьшение значений) или отсутствие тренда (значения остаются примерно на одном уровне).

- **Сезонность (Seasonality):** повторяющиеся колебания данных временного ряда, которые происходят с регулярными интервалами времени, например, ежемесячно, ежеквартально или ежегодно.
- **Цикличность (Cyclical):** колебания временного ряда, которые происходят на более длительных временных интервалах и не обязательно регулярны. Циклы могут быть связаны с экономическими, политическими или другими внешними факторами.
- **Нерегулярность/остатки (Irregularity/residuals):** представляют собой случайные и непредсказуемые изменения в данных временного ряда, которые не связаны с трендом, сезонностью или цикличностью.



Важно отметить, что не все временные ряды содержат все эти компоненты.

Для анализа необходимо сделать декомпозицию временного ряда на эти компоненты. Воспользуемся функцией `sm.tsa.seasonal_decompose` из библиотеки `statsmodels`, которая выполняет аддитивную или мультипликативную декомпозицию временного ряда на компоненты: тренд, сезонность и остатки.

Основные аргументы этой функции:

- ``series``: временной ряд, который необходимо декомпонировать.
- ``model``: вид декомпозиции, может быть 'additive' (аддитивная) или 'multiplicative' (мультипликативная).
- ``period``: период сезонности, например, 12 для ежемесячных данных или 4 для ежеквартальных.

Функция возвращает объект ``seasonal_decompose``, который содержит следующие атрибуты:

- ``trend``: тренд временного ряда.
- ``seasonal``: сезонная компонента.
- ``resid``: нерегулярная компонента / остатки.
- ``observed``: исходный временной ряд.

```
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Делаем дату индексом
oil_data.set_index('date', inplace=True)

# Декомпозиция временного ряда
decomposition = sm.tsa.seasonal_decompose(oil_data['price'], model='additive')

# Визуализация компонентов временного ряда
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 12))

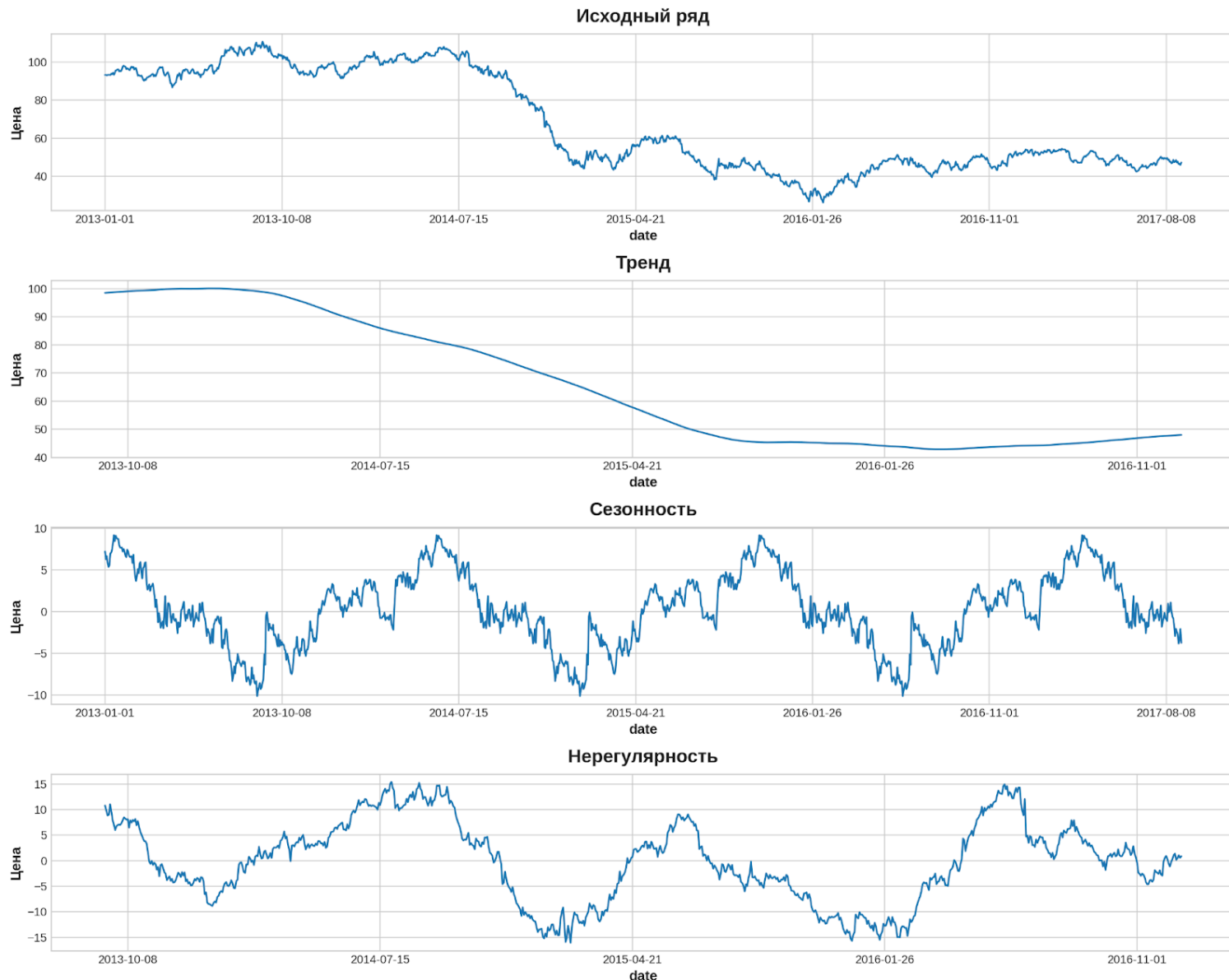
# Исходный ряд
decomposition.observed.plot(ax=ax1)
ax1.set_ylabel('Цена')
ax1.set_title('Исходный ряд')

# Тренд
decomposition.trend.plot(ax=ax2)
ax2.set_ylabel('Цена')
ax2.set_title('Тренд')

# Сезонность
decomposition.seasonal.plot(ax=ax3)
ax3.set_ylabel('Цена')
ax3.set_title('Сезонность')

# Остатки
decomposition.resid.plot(ax=ax4)
ax4.set_ylabel('Цена')
ax4.set_title('Нерегулярность')

plt.tight_layout()
plt.show()
```



Набор данных цены на нефть не совсем подходит для иллюстрации цикличности во временных рядах, так как цена на нефть не имеет ярко выраженной цикличности.

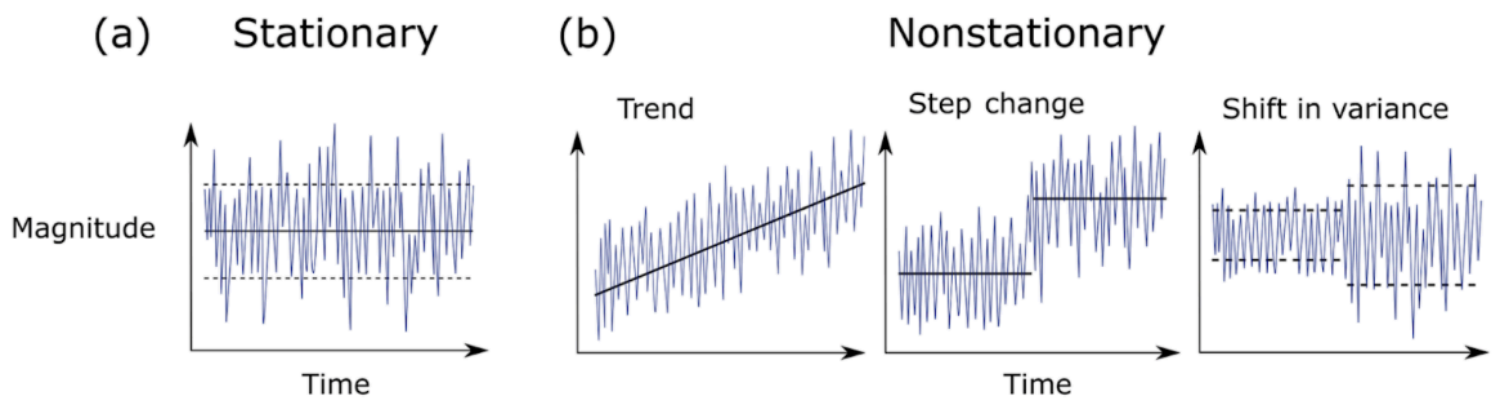
Ограничения анализа временных рядов

Анализ временных рядов, несмотря на его широкое применение, имеет ряд ограничений и сложностей, о которых важно знать. Вот некоторые из основных ограничений:

- Анализ временных рядов основан на предположении *стационарности*, что может ограничивать его применимость к нестационарным процессам.
- Пропущенные значения могут приводить к искажению результатов, особенно если они не являются случайными
- Временные ряды часто зависят от множества внешних факторов.
- Нехватка данных или слишком короткие временные ряды ограничивают возможности анализа.

Тут стоит отдельно рассмотреть два ключевых понятия: стационарные и нестационарные временные ряды.

- **Стационарные** временные ряды должны соответствовать следующим правилам:
 - Среднее значение уровней ряда не изменяется во времени, т. е. математическое ожидание постоянно.
 - Автоковариация (т. е. ковариация между уровнями одного и того же ряда) зависит только от того, насколько сильно они удалены друг от друга во времени, и не зависит от того, находятся ли они в начале или в конце временного ряда.
 - Дисперсия стационарного временного ряда также не меняется со временем.
- Если среднее значение, дисперсия или ковариация изменяются во времени, набор данных называется **нестационарным**.



Методы проверки стационарности временных рядов

На этапе подготовки модели временных рядов необходимо определить, является ли набор данных стационарным. Для этого используются статистические тесты. Существует два основных теста для проверки стационарности:

- Тест Дикки-Фуллера с поправкой на автокорреляцию (ADF-тест)
- Тест Квитка-Филлипса-Шмидта-Шина (KPSS-тест)

Тест Дикки-Фуллера с поправкой на автокорреляцию (ADF-тест)

ADF-тест использует t -статистику Дикки-Фуллера, которая рассчитывается как отношение коэффициента авторегрессии первого порядка (AR(1)) к его стандартной ошибке.

ADF-тест является наиболее распространенным статистическим тестом для проверки стационарности. Он позволяет определить, содержит ли временной ряд единичный корень, что указывает на его нестационарность. Он проводится на основе следующих гипотез:

- **Нулевая гипотеза (H_0):** ряд нестационарен, т.е. содержит единичный корень.
- **Альтернативная гипотеза (H_1):** ряд стационарен, т.е. не содержит единичного корня.

Результат теста определяется p-value:

- p-value > 0.05 - невозможно отвергнуть нулевую гипотезу (H_0). Ряд считается нестационарным.
- p-value <= 0.05 - можно принять альтернативную гипотезу (H_1). Ряд считается стационарным.

Для выполнения ADF-теста будем использовать функцию `adfuller` из модуля `stattools` библиотеки `statsmodels.tsa`.

Функция принимает следующие аргументы:

- `x` - временной ряд, для которого необходимо провести тест.
- `maxlag` - максимальное число лагов, используемое в расширенной версии теста Дики-Фуллера. Если не указано, то определяется автоматически.
- `regression` - тип регрессии, используемой в тесте: "c" (константа), "ct" (константа и тренд), "ctt" (константа, линейный и квадратичный тренд) или "nc" (без константы и тренда).
- `autolag` - метод выбора оптимального числа лагов: "AIC" (критерий Akaike), "BIC" (критерий Шварца) или "t-stat" (t-статистика).

Функция возвращает следующие значения:

- `adf_stat` - значение статистики теста Дики-Фуллера.
- `pvalue` - p-value, используемое для определения значимости результата теста.
- `usedlag` - количество лагов, использованных в тесте.
- `nobs` - число наблюдений в исходном временном ряде.
- `critical_values` - критические значения для различных уровней значимости.
- `icbest` - информационный критерий, использованный для выбора оптимального числа лагов.

```
from statsmodels.tsa.stattools import adfuller
```

```
# Исходный временной ряд
oil_data['price']
```

```
# ADF-тест
result_adf = adfuller(oil_data['price'])
print('ADF Test:')
print(f'ADF Statistic: {result_adf[0]}')
print(f'p-value: {result_adf[1]}')
print(f'Critical Values:')
for key, value in result_adf[4].items():
    print(f'    {key}: {value}')
```

Вывод:

ADF Test:

ADF Statistic: **-0.8937687985111301**

p-value: **0.790017514932319**

Critical Values:

1%: -3.435739110194116

5%: -2.863919777127088

10%: -2.5680370312770515

Поскольку значение p-value (0.790) значительно больше 0.05, мы не можем отвергнуть нулевую гипотезу H_0 о наличии единичного корня. Это означает, что временной ряд не является стационарным по результатам ADF-теста.

Проверим результат с помощью KPSS-теста.

Тест Квитка-Филлипса-Шмидта-Шина (KPSS-тест)

KPSS-тест основывается на анализе суммы квадратов отклонений накопленных частичных сумм от линейного тренда.

В отличие от ADF-теста, который проверяет нулевую гипотезу о нестационарности, KPSS-тест проверяет нулевую гипотезу о стационарности вокруг детерминированного тренда, против альтернативы наличия единичного корня. Он проводится на основе следующих гипотез:

- **нулевая гипотеза (H_0):** ряд стационарен, т.е. не содержит единичного корня.
- **альтернативная гипотеза (H_1):** ряд нестационарен, т.е. содержит единичный корень.

Результат теста определяется p-value:

- p-value > 0.05: невозможно отвергнуть нулевую гипотезу (H_0). Ряд считается стационарным.
- p-value ≤ 0.05: можно принять альтернативную гипотезу (H_1). Ряд считается нестационарным.

```
from statsmodels.tsa.stattools import kpss
```

KPSS-тест

```
result_kpss = kpss(oil_data['price'], regression='c')
print('
KPSS Test:')
print(f'KPSS Statistic: {result_kpss[0]}')
print(f'p-value: {result_kpss[1]}')
print(f'Critical Values:')
for key, value in result_kpss[3].items():
    print(f'    {key}: {value}')
```

Вывод:

KPSS Test:

KPSS Statistic: 4.696316692244311

p-value: 0.01

Critical Values:

10%: 0.347

5%: 0.463

2.5%: 0.574

1%: 0.739

Поскольку значение p-value (0.01) меньше 0.05, мы отвергаем нулевую гипотезу H_0 о стационарности временного ряда. Это означает, что временной ряд не является стационарным по результатам KPSS-теста.

Итог: оба теста (ADF и KPSS) указывают на то, что временной ряд не является стационарным. Это означает, что среднее значение, дисперсия и ковариация изменяются со временем.

Преобразование нестационарных данных в стационарные

Вкратце рассмотрим методы преобразования нестационарных данных в стационарные для эффективного моделирования временных рядов.

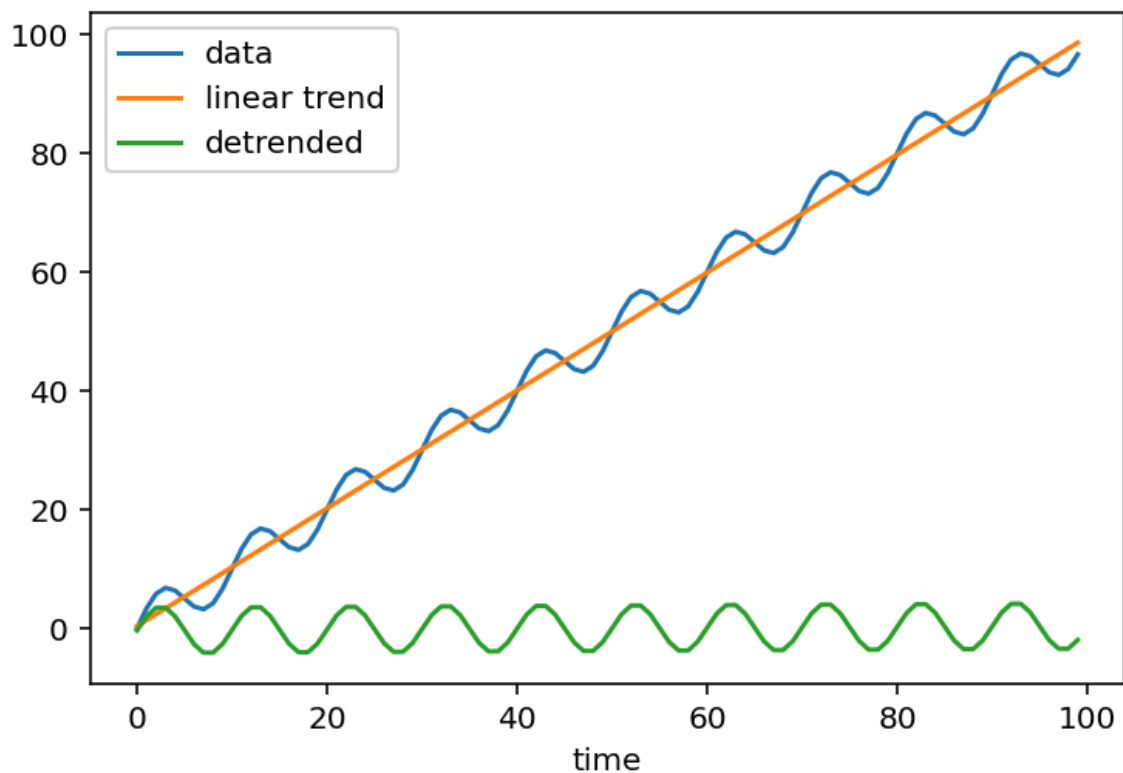
Существует три основных метода:

- Детерминирование тренда (detrending);
- Дифференцирование (differencing);
- Трансформации данных (transformation).

1. Детерминирование тренда

Этот метод заключается в удалении эффекта тренда из исходного набора данных и отображении только отклонений значений от тренда. Он позволяет выявить

циклические паттерны в данных.



Существует несколько методов детерминирования тренда:

- *Вычисление скользящего среднего* - среднего значения в определенном окне данных. Скользящее среднее позволяет сгладить тренд и выделить его основные тенденции.
- *Метод линейной регрессии*. Заключается в аппроксимации тренда с помощью линейной функции. Коэффициенты линейной функции можно найти с помощью метода наименьших квадратов.
- *Метод экспоненциального сглаживания*. Основан на идее, что более новые значения данных более информативны, чем старые. Экспоненциальное сглаживание использует весовую функцию, которая экспоненциально убывает с течением времени.
- *Метод Холта-Винтера*. Является расширением метода экспоненциального сглаживания, которое учитывает сезонность данных.

Вычислим скользящее среднее, для этого используем метод `rolling()` из `pandas`, который создает окно и вычисляет среднее значение элементов внутри этого окна.

```
# Установим окно равное 12
rolling_mean = oil_data['price'].rolling(window=12).mean()

# Детерминируем тренд
detrended = oil_data['price'] - rolling_mean

plt.figure(figsize=(14, 7))
```

```
plt.plot(oil_data['price'], label='Исходный ряд')
plt.plot(rolling_mean, label='Скользящее среднее', color='red')
plt.plot(detrended, label='Detrended', color='green')
plt.title('Детерминированный тренд')
plt.xlabel('Время')
plt.ylabel('Цена')
plt.grid(False)
plt.legend()
plt.show()
```



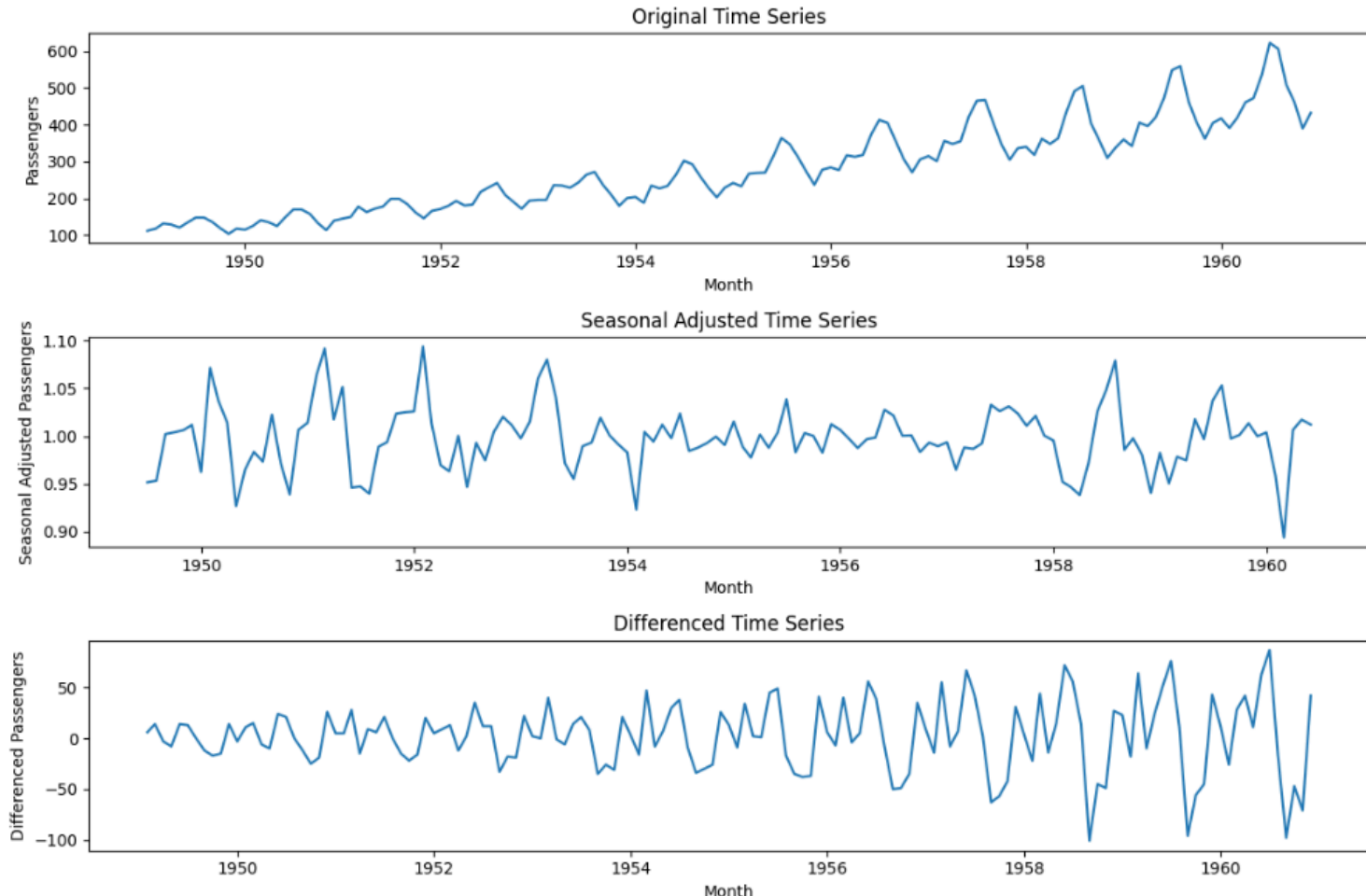
2. Дифференцирование

Дифференцирование - это простое преобразование ряда в новый временной ряд. Оно используется для устранения зависимости ряда от времени и стабилизации среднего значения ряда. В процессе дифференцирования тренд и сезонность уменьшаются. Формула такая:

$$Y_t = Y_t - Y_{t-1}$$

где,

- Y_t - значение ряда в момент времени t
- Y_{t-1} - значение ряда в предыдущий момент времени ($t-1$)

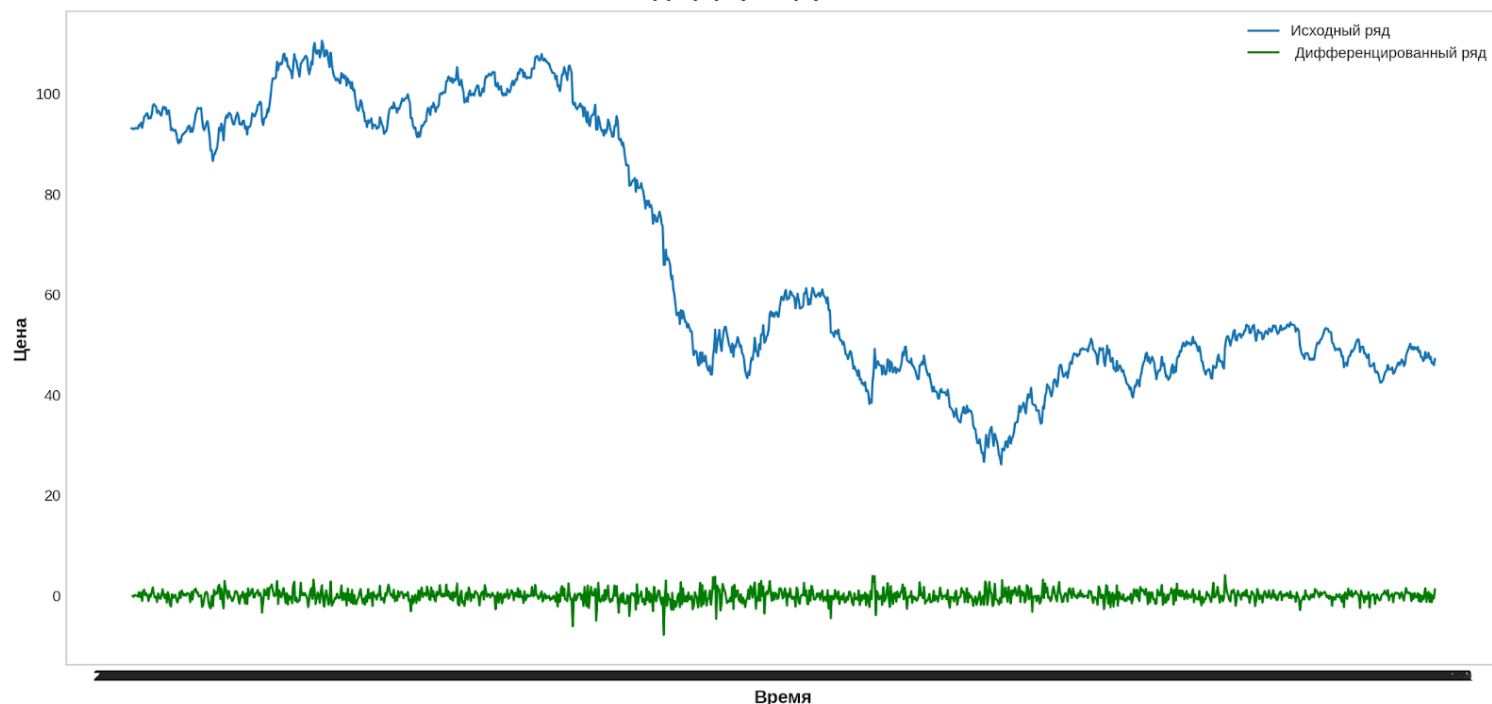


Для дифференцирования воспользуемся методом `diff()`, который вычисляет разность между текущим значением временного ряда и предыдущим значением. Это позволяет получить ряд первых разностей, который отражает изменения значений во времени.

```
differenced = oil_data['price'].diff().dropna()

plt.figure(figsize=(14, 7))
plt.plot(oil_data['price'], label='Исходный ряд')
plt.plot(differenced, label='Дифференцированный ряд', color='green')
plt.title('Дифференцирование')
plt.grid(False)
plt.legend()
plt.show()
```

Дифференцирование



3. Трансформация данных

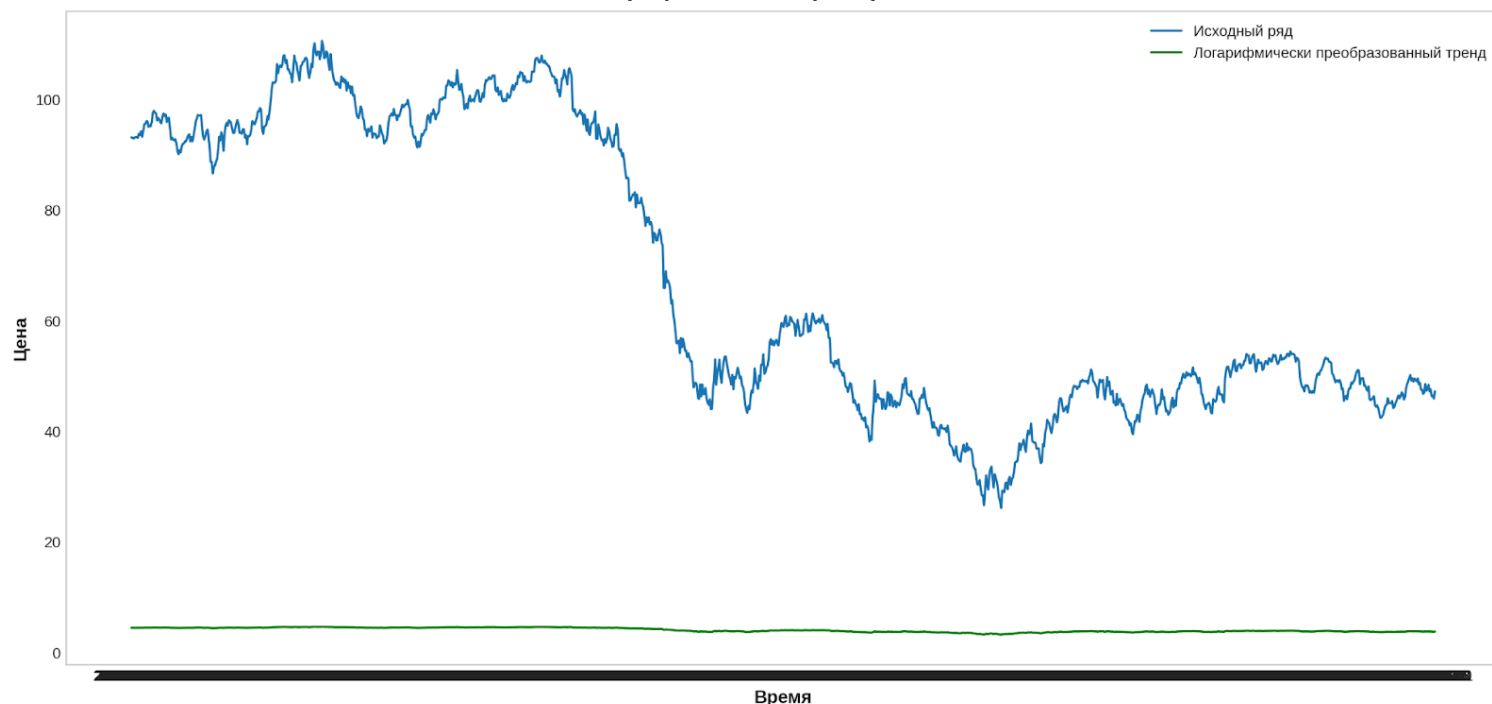
Трансформация данных включает в себя три различных метода: степенное преобразование, взятие квадратного корня и логарифмирование.

- *Степенное преобразование*: применяется для данных, сильно отклоняющихся от нуля;
- *Взятие квадратного корня*: используется для данных, имеющих значения, сгруппированные вокруг нуля;
- *Логарифмирование*: подходит для данных, резко возрастающих или колеблющихся вокруг положительных значений.

Наиболее распространенным методом является логарифмирование.

```
log_transformed = np.log(ts)
```

```
plt.figure(figsize=(14, 7))
plt.plot(ts, label='Исходный ряд')
plt.plot(log_transformed, label='Логарифмически преобразованный тренд', color='green')
plt.title('Логарифмическое преобразование')
plt.xlabel('Время')
plt.ylabel('Цена')
plt.grid(False)
plt.legend()
plt.show()
```



Алгоритмы анализа временных рядов

В предыдущих разделах мы разобрались с основами временных рядов, изучив их признаки, компоненты, ограничения и методы преобразования нестационарных данных в стационарные. Теперь перейдём к практическим инструментам, которые позволят прогнозировать, объяснять и извлекать ценную информацию из временных рядов.

В этом разделе мы рассмотрим различные алгоритмы анализа временных рядов. Но для начала загрузим и подготовим данные о продажах (`train.csv`), проведем проверку на стационарность и, при необходимости, преобразуем данные к стационарному виду.

Подготовка данных для анализа продаж

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller

# Загрузка данных
train_data = pd.read_csv('train.csv')

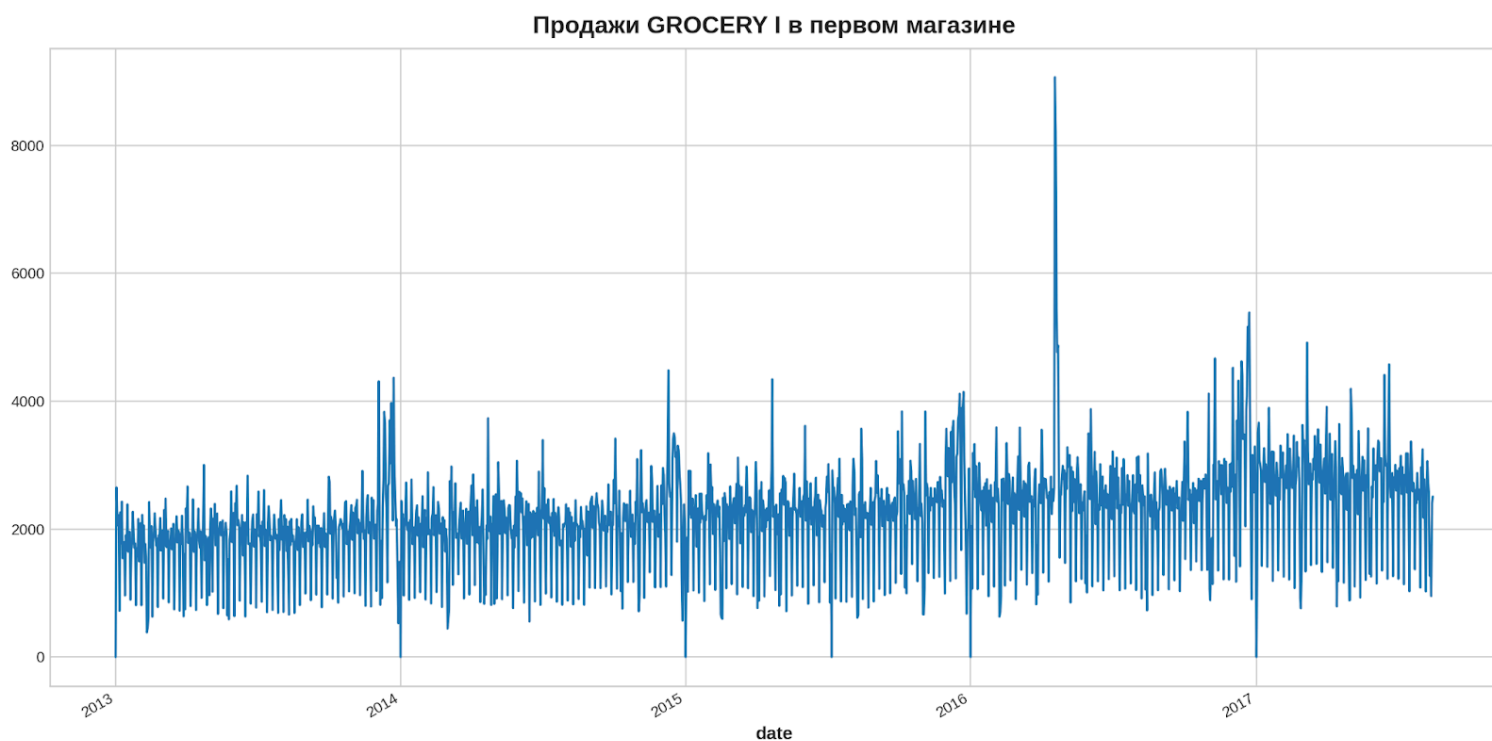
# будем анализировать продажи товаров группы GROCERY I в первом магазине
df = train_data[(train_data['family'] == 'GROCERY I') & (train_data['store_n

# Преобразование даты и установка ее в качестве индекса
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)
```

```
# Временной ряд продаж
ts = df['sales']
ts.plot(title='Sales Over Time', figsize=(14, 7))
plt.show()

# ADF-тест
result_adf = adfuller(ts)
print('ADF Test:')
print(f'ADF Statistic: {result_adf[0]}')
print(f'p-value: {result_adf[1]}')
print(f'Critical Values:')
for key, value in result_adf[4].items():
    print(f'    {key}: {value}')
```

```
ADF Test:
ADF Statistic: -4.150401045659741
p-value: 0.0007985311957746648
Critical Values:
1%: -3.4342978282123258
5%: -2.863283741222885
10%: -2.567698326213784
```



Итог: На основе результатов теста ADF, так как $p\text{-value} < 0.05$, временной ряд можно считать стационарным. Следовательно, дополнительных преобразований для стационарности не требуется, и можно сразу перейти к моделям.

Существует множество различных моделей анализа временных рядов, одни из самых часто используемых:

- **Авторегрессия (AR):** модель, которая прогнозирует значение целевой переменной на основе ее предыдущих значений.

- **Модели скользящего среднего (MA):** модель, которая прогнозирует значение целевой переменной на основе ошибок прогнозов предыдущих периодов.
- **Авторегрессионная модель скользящего среднего (ARMA):** комбинация моделей AR и MA.
- **Интегрированная авторегрессивная модель скользящего среднего (ARIMA):** Модель ARMA, которая учитывает нестационарность данных.

Авторегрессивная модель (Auto-Regressive Model, AR)

Авторегрессивная модель (AR) - это простая модель, которая предсказывает будущее значение временного ряда на основе его прошлых значений. Она часто применяется для прогнозирования, когда существует зависимость между текущим значением и несколькими предыдущими и последующими значениями ряда.

Уравнение модели авторегрессивной модели:

$$Y_t = C + b_1 Y_{t-1} + b_2 Y_{t-2} + \dots + b_p Y_{t-p} + \varepsilon_t$$

где:

- p – количество прошлых значений (лаг)
- Y_t – функция от разных прошлых значений
- ε_t – ошибки во времени (случайные отклонения)
- C – константа (свободный член)

Модель AR является по сути линейной регрессией, но использует в качестве входных данных прошлые значения того же временного ряда (с некоторым сдвигом во времени - лагом).

Для построения модели воспользуемся классом `AutoReg` из модуля `ar_model` в библиотеке `statsmodels.tsa`. Для наглядного отображения принципа построения прогнозных значений здесь и далее будем предсказывать значения на ближайшие 100 дней.

```
from statsmodels.tsa.ar_model import AutoReg

# Определение и обучение модели AR с лагом 12
model_ar = AutoReg(ts, lags=12).fit()

# Прогнозирование на 100 шагов вперед
forecast_ar = model_ar.predict(start=0, end=len(ts) + 100, dynamic=False)

# Создание индекса для прогнозного периода
```



```
forecast_index_ar = pd.date_range(start=ts.index[0], periods=len(forecast_ar
```

```
# Построение графика прогноза
```

```
plt.figure(figsize=(14, 7))
```

```
plt.plot(ts, label='Исходный ряд')
```

```
plt.plot(forecast_index_ar, forecast_ar, label='Предсказанный ряд', color='r')
```

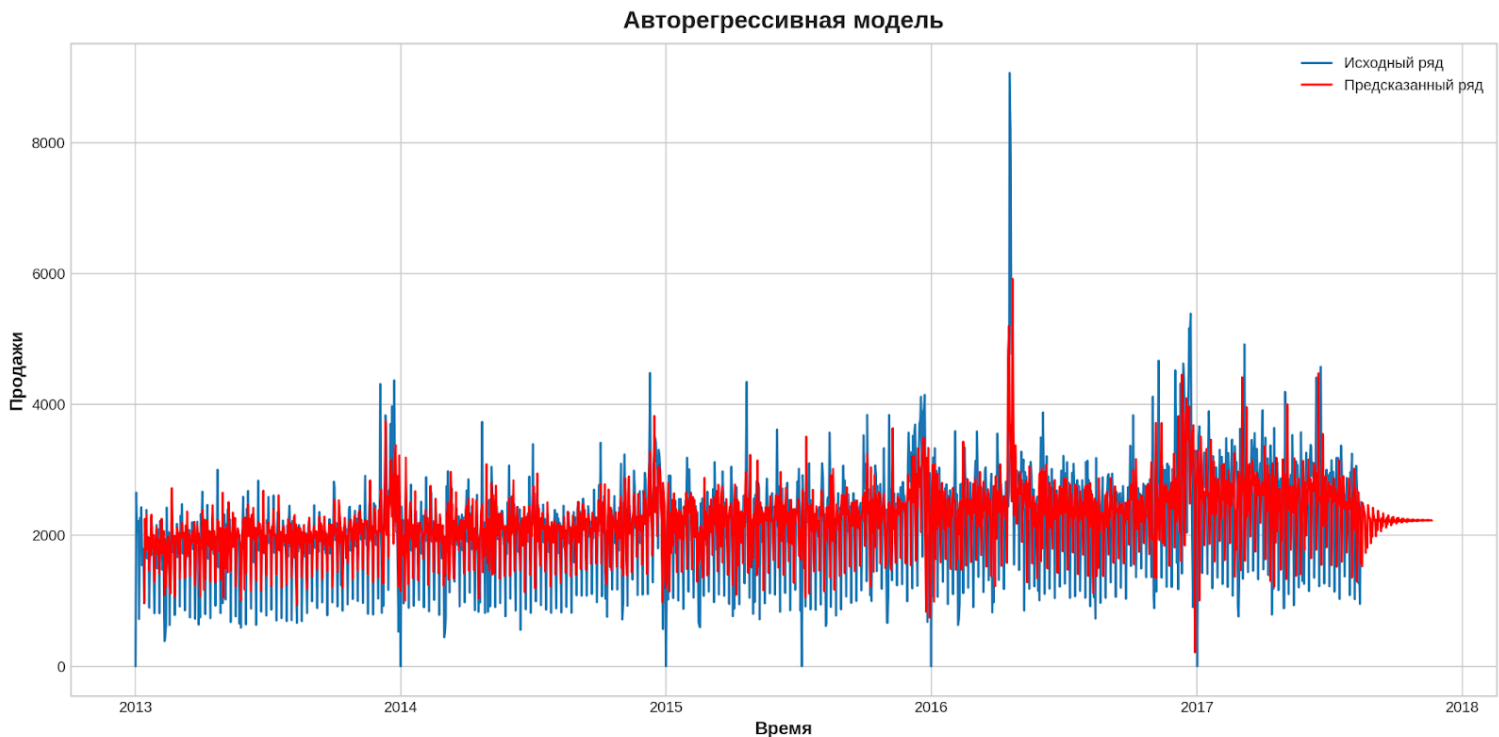
```
plt.title('Авторегрессивная модель')
```

```
plt.xlabel('Время')
```

```
plt.ylabel('Продажи')
```

```
plt.legend()
```

```
plt.show()
```



Модель скользящего среднего (Moving Average)

Скользящее среднее (Moving Average, MA) - это широко используемый метод анализа временных рядов. Он позволяет сгладить случайные краткосрочные колебания и выявить более долгосрочные тенденции в данных.

Суть метода в том, что значение скользящего среднего рассчитывается путем усреднения значений временного ряда за последние k периодов.

Существует три основных типа скользящего среднего:

1. Простое скользящее среднее (Simple Moving Average, SMA)
2. Кумулятивное скользящее среднее (Cumulative Moving Average, CMA)
3. Экспоненциальное скользящее среднее (Exponential Moving Average, EMA)

Простое скользящее среднее (SMA)

Формула простого скользящего среднего выглядит следующим образом:

$$SMA = (\sum X_i) / n$$

где:

- $\sum X_i$ – сумма значений за выбранный период времени;
- n – количество значений в периоде

SMA рассчитывает среднее арифметическое без учета весов для последних M или N точек. Выбор размера окна скользящего среднего (M или N) влияет на степень сглаживания данных:

- увеличение M или N приводит к более гладкой линии тренда, но снижает чувствительность к резким изменениям.
- уменьшение M или N делает линию тренда более чувствительной к изменениям, но при этом на ней будет больше шума.

Рассмотрим временной ряд цен акций за 5 дней: [10, 12, 11, 13, 12].

Для расчета SMA с окном $M = 3$, используем следующие шаги:

1. Первое значение SMA: $(10 + 12 + 11) / 3 = 11$
2. Второе значение SMA: $(12 + 11 + 13) / 3 = 12$
3. Третье значение SMA: $(11 + 13 + 12) / 3 = 12$

Результат: [11, 12, 12]

Этот SMA сглаживает данные, показывая общую тенденцию и устраняя случайные колебания.

Для построения простого скользящего среднего можно использовать метод `.rolling()` из `pandas`. Этот метод создает скользящее окно по данным.

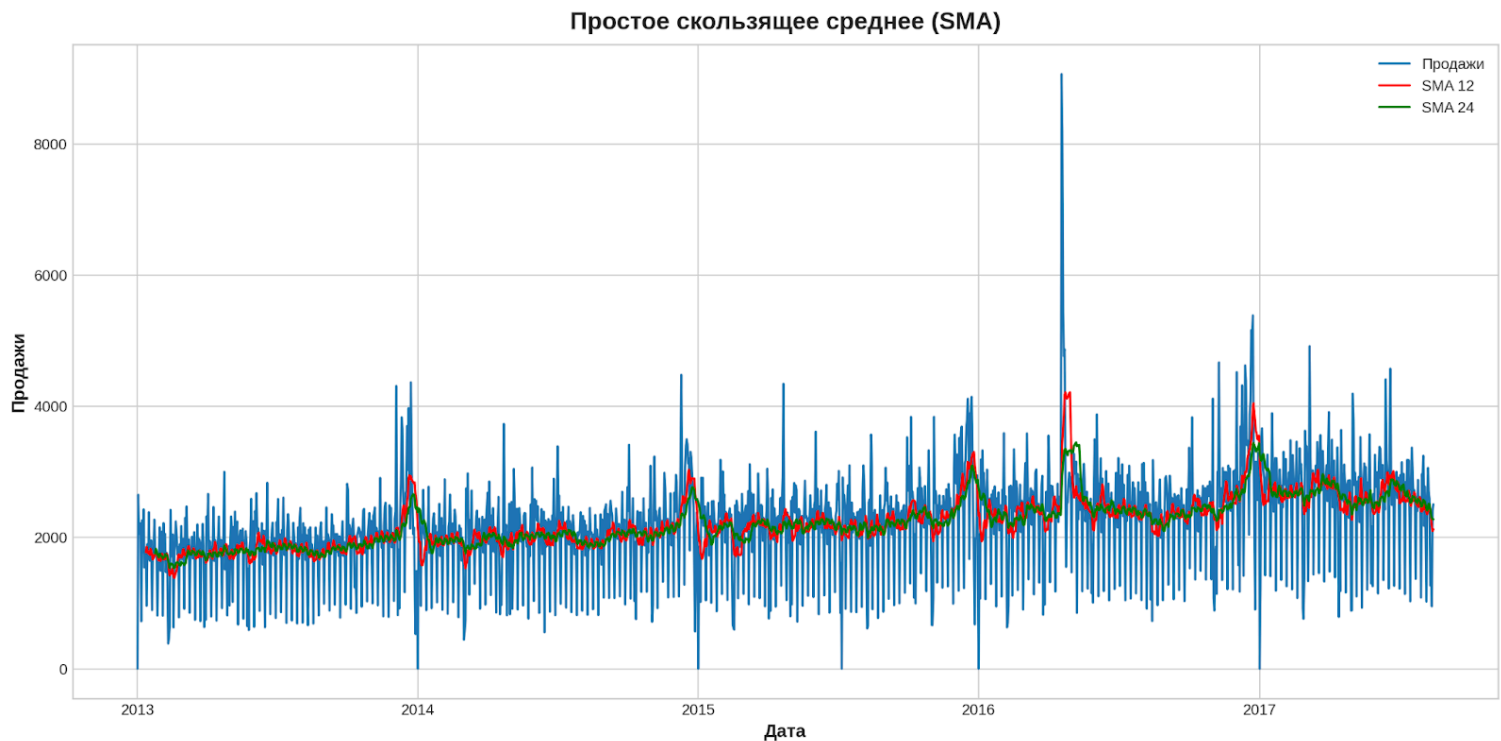
Параметр `window` задает размер этого окна.

```
# Параметры для скользящего среднего
window_size_short = 12 # короткий период
window_size_long = 24 # длинный период

# Расчет скользящего среднего
sma_short = ts.rolling(window=window_size_short).mean()
sma_long = ts.rolling(window=window_size_long).mean()

# Визуализация
plt.figure(figsize=(14, 7))
plt.plot(ts, label='Продажи')
plt.plot(sma_short, label=f'SMA {window_size_short}', color='red')
plt.plot(sma_long, label=f'SMA {window_size_long}', color='green')
plt.title('Простое скользящее среднее (SMA)')
```

```
plt.xlabel('Дата')
plt.ylabel('Продажи')
plt.legend()
plt.show()
```



Если с помощью МА необходимо предсказать будущий тренд, то можно воспользоваться циклом.

```
# Параметры для скользящего среднего
window_size = 12

# Расчет скользящего среднего
sma = ts.rolling(window=window_size).mean()

# Прогнозирование на 100 дней вперед
forecast_days = 100
last_known_date = ts.index[-1]

# Создание будущих дат
future_dates = [last_known_date + pd.Timedelta(days=i) for i in range(1, for

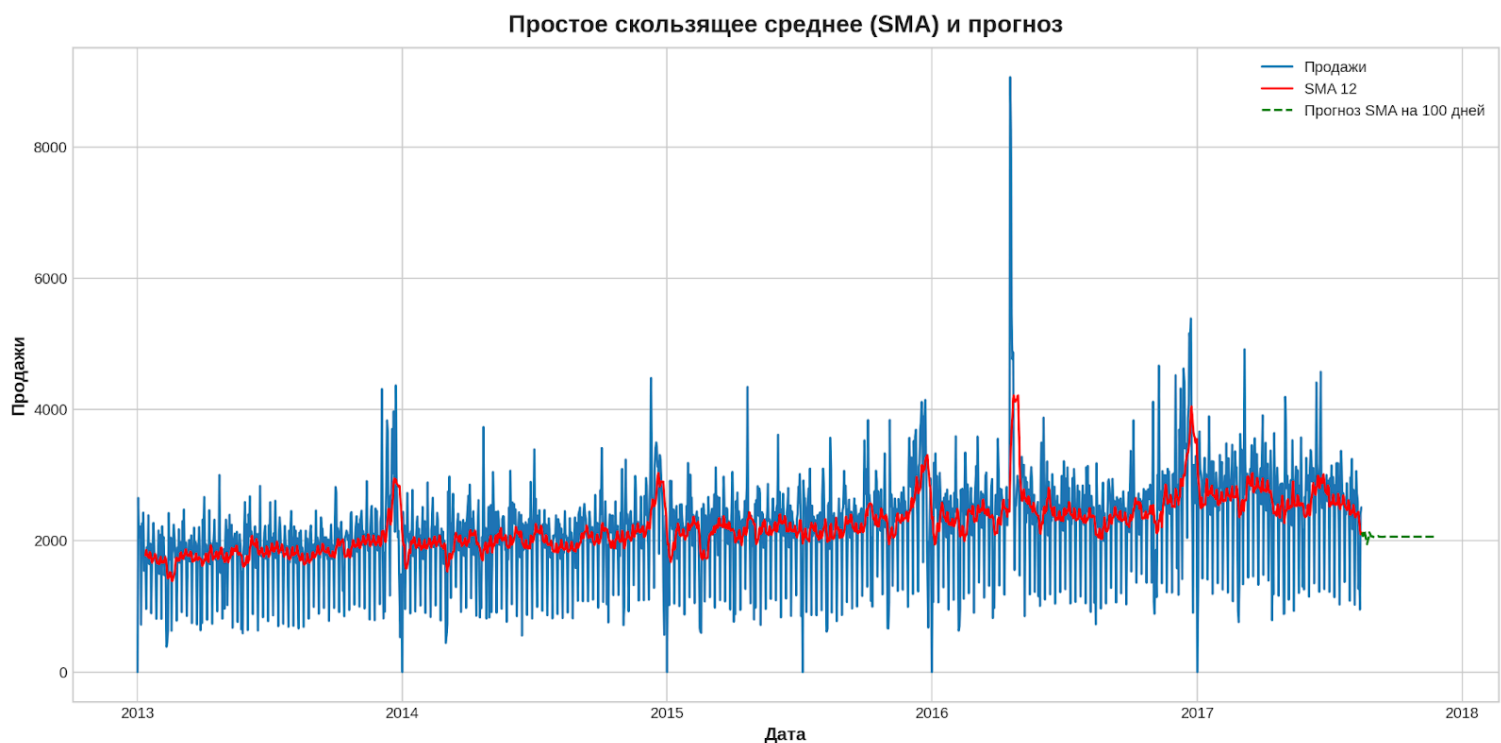
# Расчет будущего значения SMA
future_sma = []
last_values = ts[-window_size:].tolist() # последние известные значения, ис
for i in range(forecast_days):
    future_value = np.mean(last_values)
    future_sma.append(future_value)
    last_values.pop(0)
    last_values.append(future_value)

# Создание DataFrame для прогнозируемых значений
future_df = pd.DataFrame(data=future_sma, index=future_dates, columns=['Fore
```

```
# Объединение исходных и прогнозируемых данных
combined_df = pd.concat([ts, future_df])
```

```
# Визуализация
```

```
plt.figure(figsize=(14, 7))
plt.plot(ts, label='Продажи')
plt.plot(sma, label=f'SMA {window_size}', color='red')
plt.plot(future_df, label='Прогноз SMA на 100 дней', color='green', linestyle='dashed')
plt.title('Простое скользящее среднее (SMA) и прогноз')
plt.xlabel('Дата')
plt.ylabel('Продажи')
plt.legend()
plt.show()
```



Кумулятивное скользящее среднее (СМА)

Кумулятивное скользящее среднее (СМА) представляет собой невзвешенное среднее значений до текущего времени. В отличие от простого скользящего среднего (SMA), которое учитывает фиксированное количество последних значений, СМА включает все данные, накопленные с начала наблюдений. ▶

Формула для расчета СМА в момент времени t выглядит следующим образом:

$$\text{СМА} = (x_1 + x_2 + \dots + x_t) / t$$

где:

- $x_1, x_2, \dots, x_t$ – значения временного ряда до текущего момента времени t ;

- t – текущее количество значений в ряду

СМА имеет следующие особенности:

1. *Плавное сглаживание*: поскольку СМА учитывает все предыдущие данные, его изменения происходят плавно и постепенно.
2. *Уменьшение влияния шума*: с увеличением количества наблюдений влияние случайных колебаний на СМА уменьшается, что делает его полезным для выявления долгосрочных трендов.
3. *Память всех значений*: в отличие от SMA, который учитывает только последние N значений, СМА учитывает все предыдущие значения, что может быть полезно для анализа данных с долгосрочными тенденциями.

Рассмотрим временной ряд цен акций за 5 дней: [10, 12, 11, 13, 12].

1. Первое значение СМА: $(10) / 1 = 10$
2. Второе значение СМА: $(10 + 12) / 2 = 11$
3. Третье значение СМА: $(10 + 12 + 11) / 3 = 11$
4. Четвертое значение СМА: $(10 + 12 + 11 + 13) / 4 = 11.5$
5. Пятое значение СМА: $(10 + 12 + 11 + 13 + 12) / 5 = 11.6$

Результат: [10, 11, 11, 11.5, 11.6]

Таким образом, СМА помогает сглаживать данные и выявлять общие тенденции, уменьшая влияние случайных колебаний и шума.

В Python для расчёта СМА можно использовать метод `.cumsum()` из Pandas.

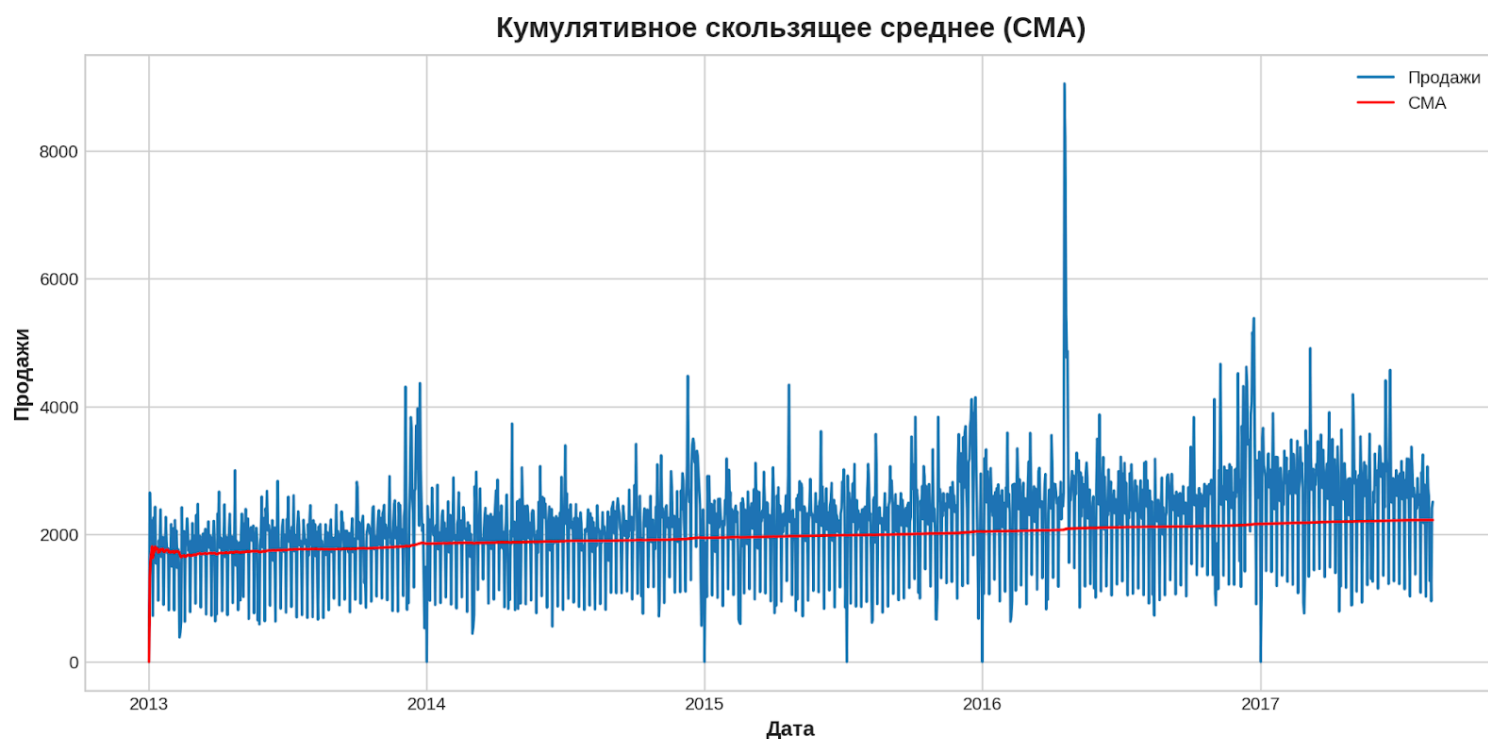
```
# Рассчитываем кумулятивное скользящее среднее
cma = ts.cumsum() / (pd.Series(range(1, len(ts) + 1), index=ts.index))

# Вывод результатов
print(cma)

# Визуализация
plt.figure(figsize=(12, 6))
plt.plot(ts, label='Продажи')
plt.plot(cma, label='СМА', color='red')
plt.title('Кумулятивное скользящее среднее (СМА)')
plt.xlabel('Дата')
plt.ylabel('Продажи')
plt.legend()
plt.show()

# Вывод:
```

2013-01-01	0.000000
2013-01-02	1326.000000
2013-01-03	1591.000000
2013-01-04	1707.250000
2013-01-05	1809.000000
...	
2017-08-11	2224.003571
2017-08-12	2223.650208
2017-08-13	2222.894174
2017-08-14	2223.003565
2017-08-15	2223.172803



Экспоненциальное скользящее среднее (ЕМА)

ЕМА используется в основном для определения трендов и фильтрации шума. Веса элементов уменьшаются постепенно со временем, что означает, что большее значение придается недавним точкам данных, а не историческим. По сравнению с SMA, ЕМА быстрее реагирует на изменения и более чувствителен к ним.

Формула для расчета ЕМА:

$$EMA_t = \alpha * x_t + (1 - \alpha) * EMA_{\{t-1\}}$$

где:

- EMA_t — текущее значение экспоненциального скользящего среднего
- x_t — текущее значение временного ряда
- α — коэффициент сглаживания. Имеет значение между 0 и 1, представляет вес, применяемый к самым недавним периодам.

- $EMA_{\{t-1\}}$ – предыдущее значение экспоненциального скользящего среднего

Рассмотрим временной ряд цен акций за 5 дней: [10, 12, 11, 13, 12].

Для $\alpha = 0.1$:

1. Первое значение ЕМА (EMA_1) = 10 (начальное значение совпадает с первым значением ряда)
2. $EMA_2 = 0.1 * 12 + 0.9 * 10 = 10.2$
3. $EMA_3 = 0.1 * 11 + 0.9 * 10.2 = 10.28$
4. $EMA_4 = 0.1 * 13 + 0.9 * 10.28 = 10.55$
5. $EMA_5 = 0.1 * 12 + 0.9 * 10.55 = 10.7$

Для $\alpha = 0.3$:

1. Первое значение ЕМА (EMA_1) = 10 (начальное значение совпадает с первым значением ряда)
2. $EMA_2 = 0.3 * 12 + 0.7 * 10 = 10.6$
3. $EMA_3 = 0.3 * 11 + 0.7 * 10.6 = 10.82$
4. $EMA_4 = 0.3 * 13 + 0.7 * 10.82 = 11.55$
5. $EMA_5 = 0.3 * 12 + 0.7 * 11.55 = 11.78$

Таким образом, ЕМА с разными значениями коэффициента сглаживания показывает, как недавние данные влияют на среднее значение, делая его более чувствительным к последним изменениям.

В Python для расчета экспоненциального скользящего среднего (ЕМА) можно использовать метод `.ewm()` из библиотеки Pandas.

```
# Коэффициент сглаживания
```

```
alpha = 0.2
```

```
# Рассчитываем экспоненциальное скользящее среднее
```

```
ema = ts.ewm(alpha=alpha, adjust=False).mean()
```

```
# Визуализация
```

```
plt.figure(figsize=(12, 6))
```

```
plt.plot(ts, label='Продажи')
```

```
plt.plot(ema, label=f'ЕМА (alpha={alpha})', color='red')
```

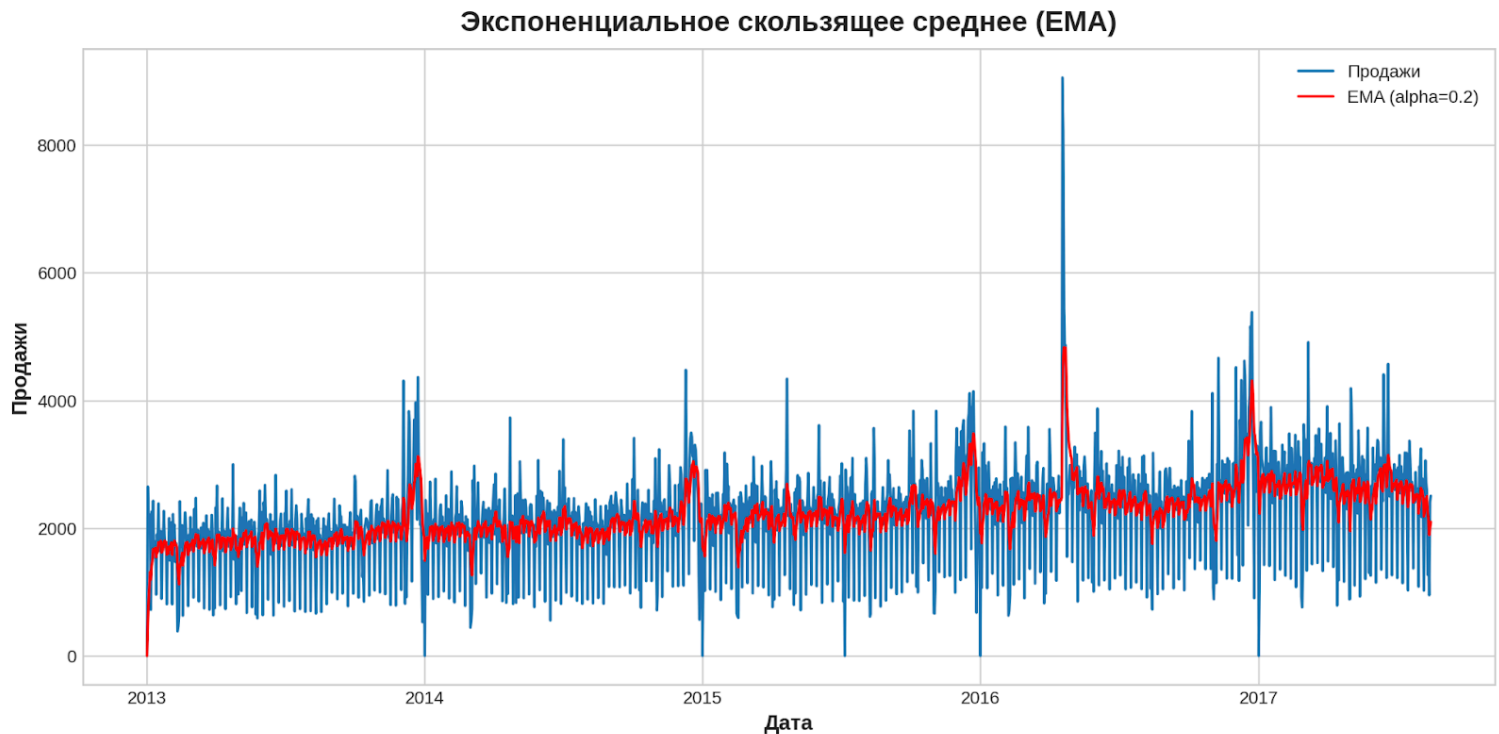
```
plt.title('Экспоненциальное скользящее среднее (ЕМА)')
```

```
plt.xlabel('Дата')
```

```
plt.ylabel('Продажи')
```



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



Модели ARMA и ARIMA

Autoregressive Moving Average (ARMA) – модель авторегрессии-скользящего среднего. Она представляет собой комбинацию авторегрессивной модели и модели скользящего среднего. Эта модель предоставляет слабостационарный случайный процесс в терминах двух многочленов: одного для авторегрессии и второго для скользящего среднего.

Autoregressive Integrated Moving Average (ARIMA) – модель авторегрессионной интегрированной скользящей средней.

Одно из главных различий между ARMA и ARIMA заключается в интегрировании. ARIMA моделирует разности между последовательными точками данных, что делает ее способной работать с нестационарными рядами, в то время как ARMA работает с стационарными рядами без необходимости дифференцирования.

- AR $\Rightarrow$ Использует прошлые значения для прогнозирования будущего.
- MA $\Rightarrow$ Использует прошлые значения ошибок в данном ряду для прогнозирования будущего.
- I $\Rightarrow$ Использует разности наблюдений, чтобы сделать данные стационарными.
- AR + I + MA = ARIMA

ARMA лучше всего подходит для прогнозирования стационарных рядов. Для поддержки как стационарных, так и нестационарных рядов была разработана модель ARIMA.

Модели ARMA и ARIMA определяются тремя параметрами:

- p (autoregressive lags): Количество прошлых значений временного ряда, которые используются для прогнозирования текущего значения.
- q (moving average lags): Количество ошибок прогноза прошлых периодов, которые учитываются при формировании нового прогноза.
- d (difference in the order): Число раз, которое необходимо продифференцировать данные, чтобы сделать их стационарными (т.е. устранить тренд и сезонность). В модели ARMA порядок разностей обычно равен нулю, поскольку ARMA не требует дифференцирования данных.

Для определения параметров этих моделей полезно анализировать ACF и PACF.

- *Функция автокорреляции (Auto-Correlation Function, ACF);*
- *Частичная функция автокорреляции (Partial Auto-Correlation, PACF).*

ACF измеряет степень сходства между значением временного ряда и его прошлыми значениями с разными временными лагами. Другими словами, ACF измеряет корреляцию между временными отклонениями ряда и его отставаниями на различные лаги.

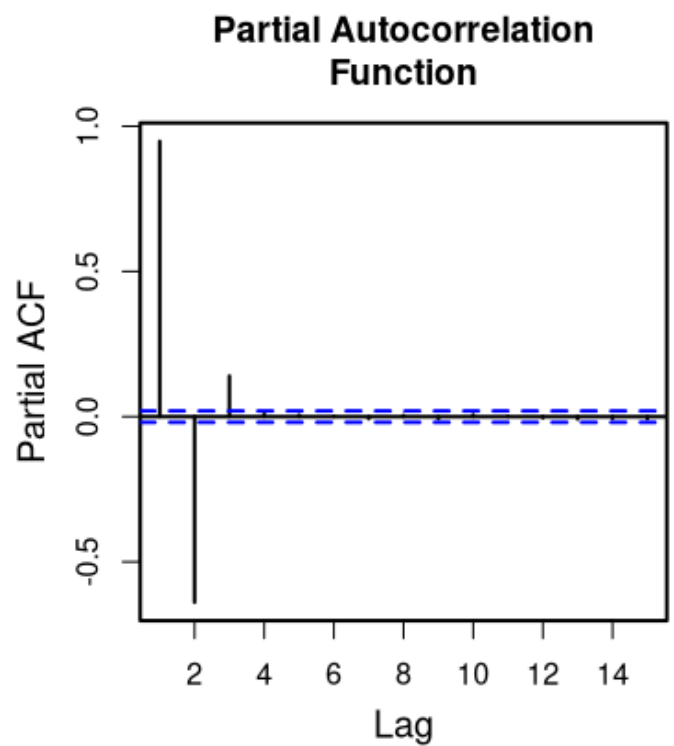
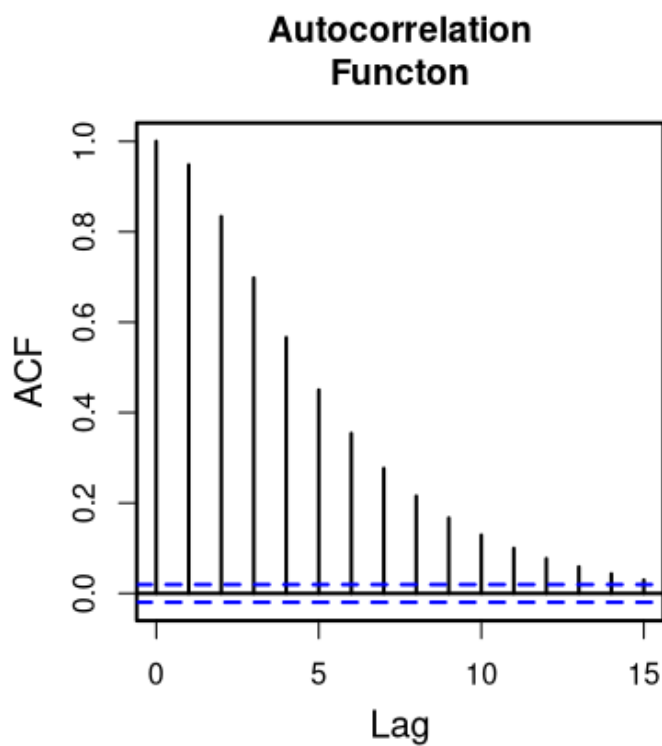
Корреляция, близкая к 1 или -1, указывает на сильную зависимость между значениями на различных временных точках, в то время как корреляция, близкая к 0, указывает на отсутствие зависимости.

На практике ACF позволяет определить наличие сезонных или циклических паттернов во временном ряду.

PACF похожа на функцию автокорреляции, но немного сложнее для понимания. PACF показывает прямую корреляцию между текущим значением и его прошлыми значениями с учетом временных лагов. При этом она исключает влияние промежуточных значений на эту корреляцию.

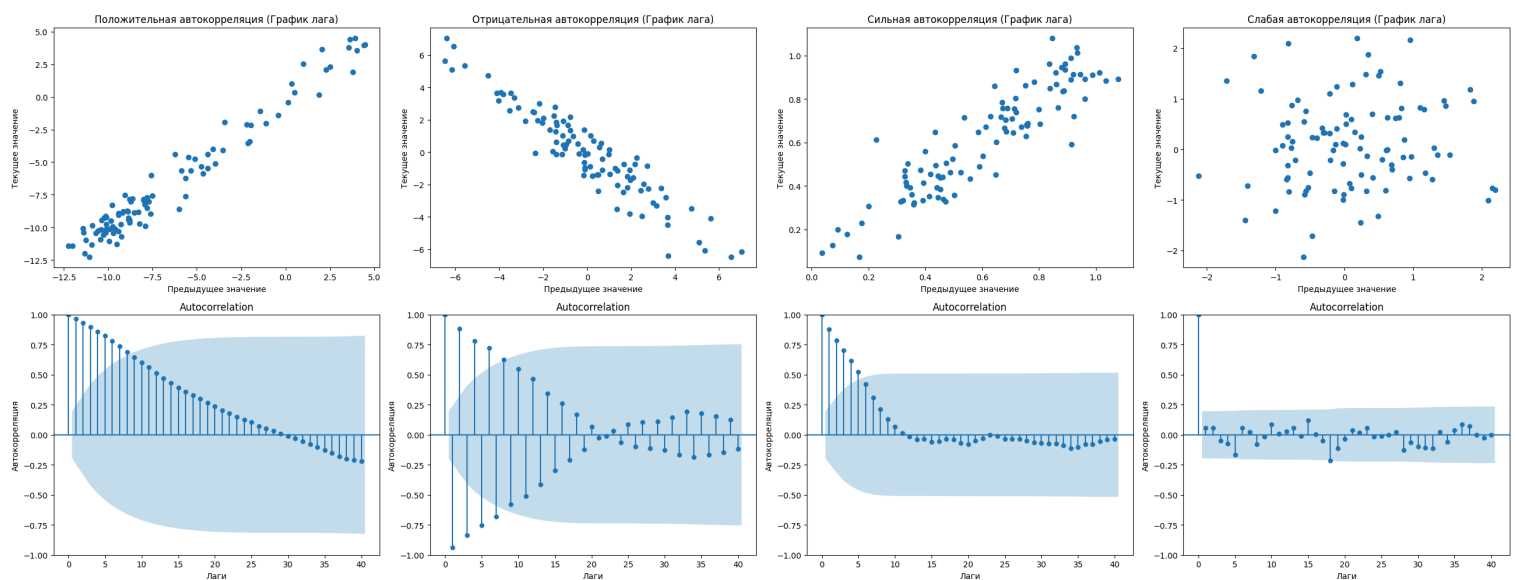
Другими словами, PACF фокусируется только на непосредственном влиянии прошлых значений на текущее, игнорируя косвенное влияние, которое может идти через другие промежуточные значения.

PACF часто используется при определении порядка авторегрессионной части (AR) модели ARMA и ARIMA, помогая определить количество лагов, которые следует включить в модель AR.



Изучение функций автокорреляции и частичной автокорреляции является важной частью анализа временных рядов и выбора подходящих моделей для прогнозирования.

Типы автокорреляции



Интерпретация графиков ACF и PACF

ACF	PACF	Подходящая модель
Постепенное убывание	Мгновенное падение	Авторегрессии (AR)
Мгновенное падение	Постепенное убывание	Скользящее среднее (MA)
Постепенное убывание	Постепенное убывание	ARMA / ARIMA
Мгновенное падение	Мгновенное падение	Ни одна модель не подходит

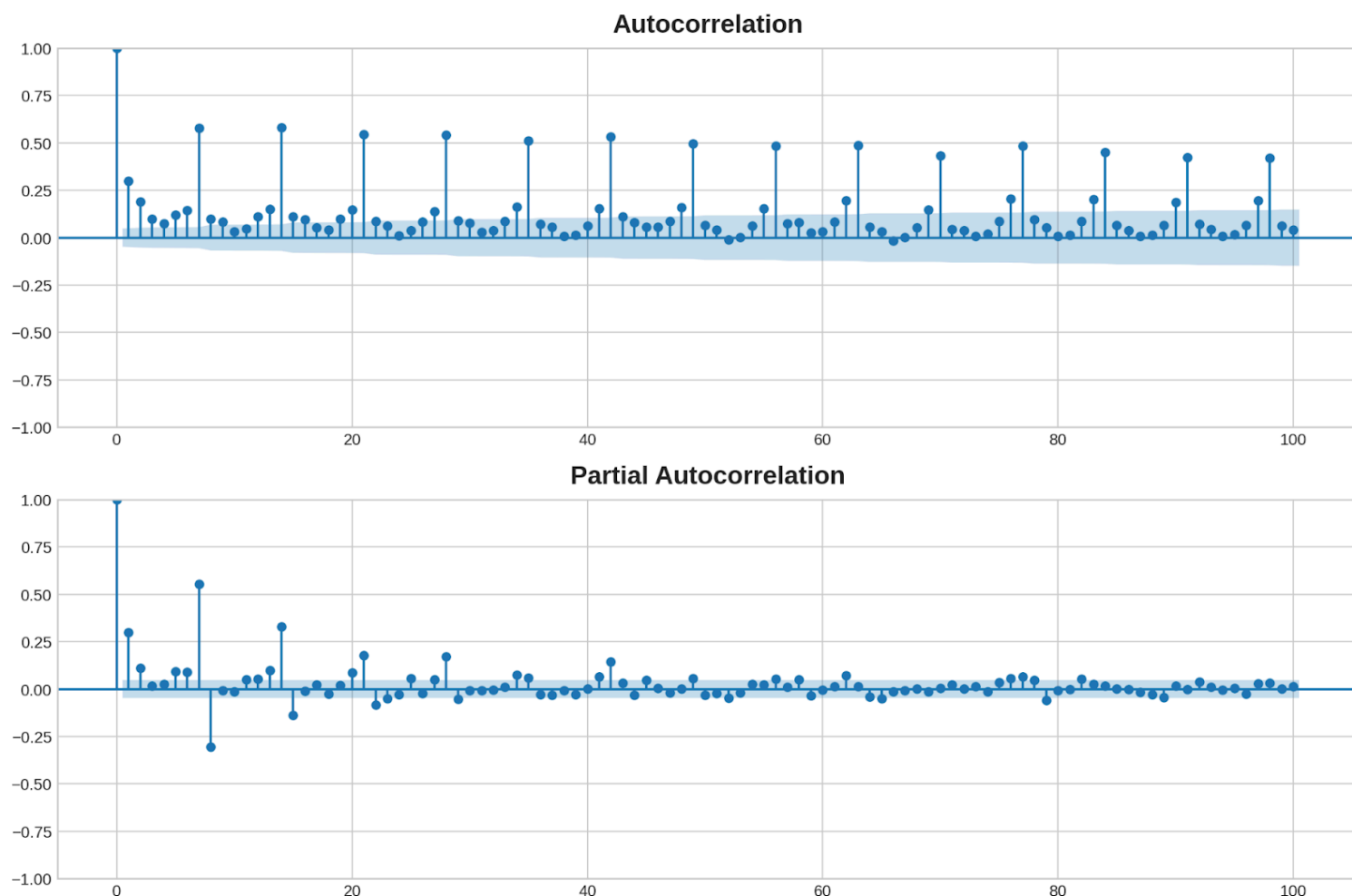
- *Авторегрессионная часть (AR)*: если ACF убывает медленно и PACF обрывается после нескольких лагов, это может указывать на необходимость включения авторегрессионной части модели. Порядок этой части (число лагов) можно определить на основе PACF.
- *Скользящая средняя часть (MA)*: если PACF убывает медленно и ACF обрывается после нескольких лагов, это может указывать на необходимость включения скользящей средней части модели. Порядок этой части также можно определить на основе ACF.

Анализ ACF и PACF помогает идентифицировать оптимальные значения параметров для моделей ARMA и ARIMA, что улучшает ее способность к прогнозированию и адекватному описанию временного ряда.

Для анализа автокорреляционной функции и частичной автокорреляционной функции в Python можно использовать функции `plot_acf()` и `plot_pacf()` из `statsmodels`.

```
# Анализ ACF и PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

sm.graphics.tsa.plot_acf(ts, ax=ax1, lags=100)
sm.graphics.tsa.plot_pacf(ts, ax=ax2, lags=100)
plt.show()
```



На **ACF-графике** видны значительные пики на лагах 1, 5, 10, 15 и так далее, что указывает на сезонность в данных с периодом около 5 лагов. Пики на промежуточных лагах (например, на лагах 2, 3, 4 и так далее) также значимы, что может указывать на сложную структуру автокорреляции, включающую несколько компонентов. Частичная автокорреляционная функция (PACF).

На **PACF-графике** видно несколько значимых пиков на первых лагах (например, лаги 1 и 5). Эти пики постепенно уменьшаются, что указывает на присутствие AR-компоненты (авторегрессии). Значимые пики также наблюдаются на более высоких лагах (например, на лагах около 5 и далее), что дополнительно подтверждает сезонность.

Выводы и подбор параметров:

1. Параметры AR (p):

- На PACF-графике значимые пики на первых нескольких лагах (например, 1 и 5), что может указывать на AR-компоненты. Параметр p может быть около 5 или 10.

2. Параметры MA (q):

- На ACF-графике видны значительные пики на нескольких лагах, что указывает на наличие MA-компоненты. Параметр q может быть около 5 или 10.

3. Сезонность:

- Значительные пики на ACF и PACF на интервалах около 5 лагов указывают на сезонность в данных. Это можно учитывать в модели ARIMA, добавив сезонные параметры.

Для построения модели ARMA и ARIMA в Python можно воспользоваться модулем `statsmodels.tsa.arima_model`, который содержит класс ARIMA для построения моделей временных рядов.

Методы класса ARIMA:

- `fit()`: оценивает параметры модели.
- `predict()`: делает прогноз на основе модели.
- `forecast()`: выполняет прогнозирование на заданное количество шагов вперед.
- `summary()`: выводит сводку результатов модели.

Воспользуемся моделью ARMA, так как у нас стационарный временной ряд.

```
# Оценка модели ARMA
p = 10 # Порядок авторегрессии (AR)
d = 0  # Порядок разностей (d)
```

```

q = 10 # Порядок скользящего среднего (MA)
model_arima = sm.tsa.ARIMA(ts, order=(p, d, q)).fit()

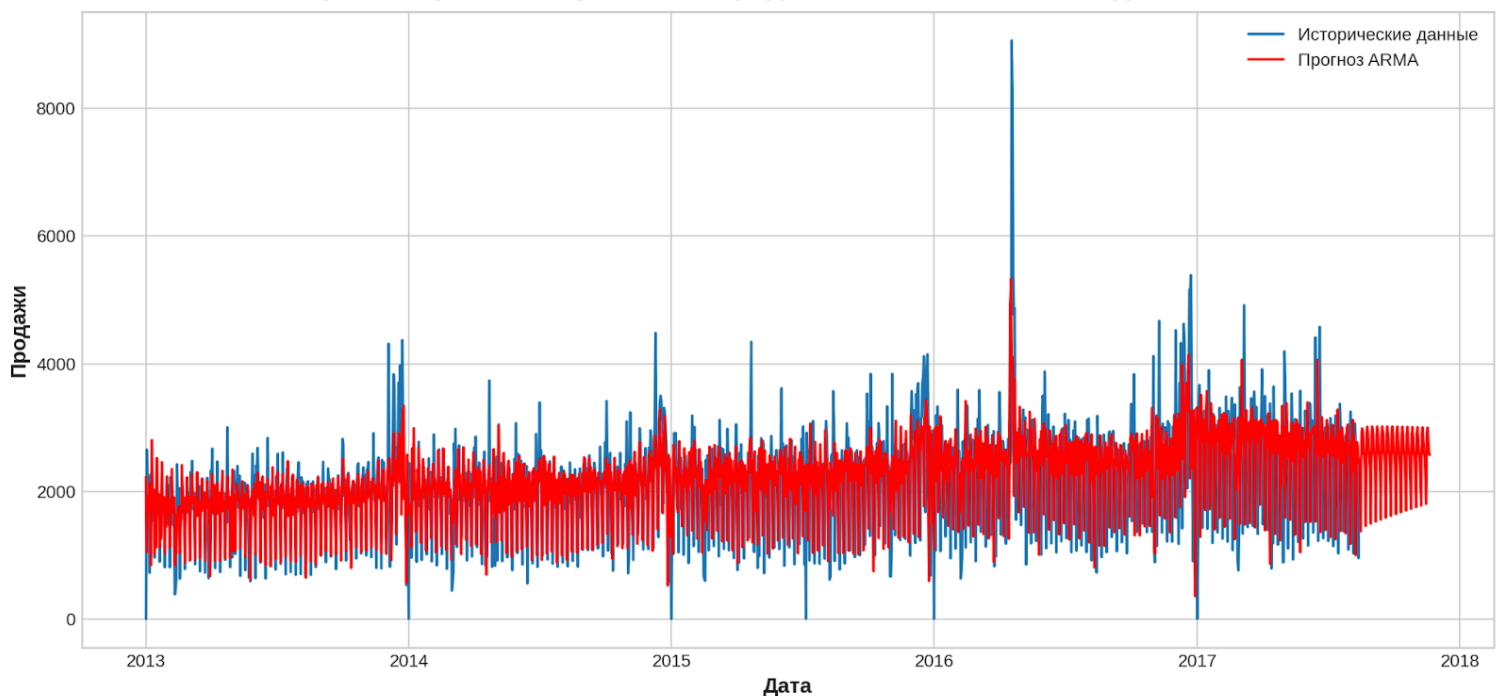
# Прогноз на 100 дней вперед
forecast_horizon = 100
forecast_arima = model_arima.predict(start=0, end=len(ts) + forecast_horizon)

# Создание индекса для прогнозного периода
forecast_index_arima = pd.date_range(start=ts.index[0], periods=len(forecast_arima))

# Визуализация исторических данных и прогноза
plt.figure(figsize=(12, 6))
plt.plot(ts.index, ts, label='Исторические данные')
plt.plot(forecast_index_arima, forecast_arima, label='Прогноз ARIMA', color='red')
plt.title('Прогнозирование временного ряда с использованием модели ARMA')
plt.xlabel('Дата')
plt.ylabel('Продажи')
plt.legend()
plt.show()

```

Прогнозирование временного ряда с использованием модели ARMA



На самом деле все ранее рассмотренные модели могут быть реализованы в Python с помощью класса ARIMA. Посмотрим на параметр `order=(p, d, q)` и возможные комбинации переменных:

- $(1, 0, 0)$: Это модель с одним предыдущим значением временного ряда для авторегрессии и без скользящего среднего (MA). То есть это Авторегрессионная модель AR(1).
- $(0, 0, 1)$: Это модель без авторегрессии и с одним предыдущим значением ошибки прогнозирования для скользящего среднего. То есть это модель Скользящего среднего MA(1).

- $(1, 0, 1)$: Это модель с одним предыдущим значением временного ряда для авторегрессии и одним предыдущим значением ошибки прогнозирования для скользящего среднего. То есть это комбинированная модель ARMA(1,1).
- $(1, 1, 1)$: Это модель ARIMA(1, 1, 1) с одним предыдущим значением временного ряда для авторегрессии, одним предыдущим значением ошибки прогнозирования для скользящего среднего и одним дифференцированием временного ряда.

Важно отметить, что выбор модели прогнозирования зависит от специфики временного ряда и наличия в нем различных компонентов. Перед применением любой модели необходимо стабилизировать временные ряды.

Сглаживание данных методами скользящего среднего позволяет уменьшить влияние случайных колебаний и выявить долгосрочные тренды. Декомпозиция временных рядов на тренд, сезонность и остаточные компоненты улучшает понимание их структуры.

Такой комплексный подход обеспечивает более точные прогнозы и способствует принятию обоснованных решений на основе полученных данных, делая анализ временных рядов эффективным инструментом для различных практических применений.

Однако важно помнить: *не стоит слепо доверять результатам прогнозирования, так как они всегда содержат некоторую степень погрешности.*