КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Р.А. БУРНАШЕВ

ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ И МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОСТРОЕНИЯ УСТОЙЧИВЫХ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Учебно-методическое пособие

для студентов бакалавриата и магистратуры, изучающих основы машинного обучения, разработку систем поддержки принятия решений

КАЗАНЬ

2025

УДК 004.891 + 004.896(075.8)
ББК: 22.1+ 22.3я73
Б91

Печатается по рекомендации учебно-методической комиссии
Института вычислительной математики и информационных технологий
Казанского (Приволжского) федерального университета
(протокол № 2 от 19 сентября 2025 г.)

Рецензенты

кандидат экономических наук, доцент, доцент кафедры анализа данных и технологий программирования Института вычислительной математики и информационных технологий Казанского (Приволжского) федерального университета **Г.З. Вахитов**.

доктор физико-математических наук, доцент, заведующий кафедры высшей математики Казанского государственного энергетического университета **А.С. Ситдиков**.

Бурнашев Р.А.

Б91 Технологии анализа данных и машинного обучения для построения устойчивых систем поддержки принятия решений: учебно-методическое пособие / Р.А. Бурнашев. — Казань: Издательство Казанского университета, 2025. — 32 с.

Учебно-методическое пособие предназначено для проведения лекционных, лабораторных и практических занятий по дисциплинам: «Введение в экспертные системы», «Основы машинного обучения» для обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика» и магистратуры 01.04.02 «Прикладная математика и информатика».

Работа выполнена за счет гранта Академии наук Республики Татарстан, предоставленного молодым кандидатам наук (постдокторантам) с целью защиты докторской диссертации, выполнения научно-исследовательских работ, а также выполнения трудовых функций в научных и образовательных организациях Республики Татарстан в рамках Государственной программы Республики Татарстан «Научно-технологическое развитие Республики Татарстан» (Соглашение от 16.12.2024 № 04/2024-ПД).

УДК 004.891 + 004.896(075.8)
ББК: 22.1+ 22.3я73

Содержание

ВВЕДЕНИЕ

Актуальность

В современном мире, переполненном данными, ключевое преимущество получают те, кто умеет не просто собирать информацию, но и извлекать из нее скрытые закономерности и строить точные прогнозы. Машинное обучение и продвинутый статистический анализ стали неотъемлемыми инструментами в арсенале экономиста, аналитика, маркетолога и исследователя. Однако, путь от сырых данных к обоснованному решению требует владения целым комплексом технологий — от базовой обработки до сложных вероятностных моделей.

О пособии

Данное учебно-методическое пособие представляет собой последовательное и практико-ориентированное введение в технологии анализа данных и машинного обучения, ориентированное на применение в системах поддержки принятия решений (СППР).

Логика изложения и связь между блоками

Пособие выстроено по принципу «от простого к сложному», где каждая последующая глава логически вытекает из предыдущей:

Глава 1 закладывает фундамент, обучая студентов работе с данными — их первичной обработке, анализу временных рядов и визуализации. Это базовый навык для любого аналитика.

Глава 2 углубляется в анализ, знакомя с ключевыми статистическими концепциями (стационарность) и моделями (регрессия). Это мост между описательной статистикой и прогнозным моделированием.

Глава 3 посвящена продвинутым методам анализа зависимостей. Здесь показывается, как выйти за рамки линейной корреляции с помощью копул, что критически важно для оценки рисков в финансах и экономике.

Глава 4 знакомит с одним из самых мощных алгоритмов машинного обучения — градиентным бустингом. Студенты не только поймут его теорию, но и реализуют алгоритм "вручную", что обеспечивает глубокое понимание его работы, а затем применят готовую библиотеку для решения прикладных задач.

Целевая аудитория

Пособие предназначено для студентов бакалавриата и магистратуры направлений «Прикладная информатика», «Прикладная математика и информатика», а также для всех, кто хочет получить структурированные и прикладные знания в области машинного обучения и анализа данных.

Результат освоения

Освоив материалы учебно-методического пособия, студент будет способен самостоятельно проводить анализ данных: от предобработки и разведочного анализа до построения и верификации статистических моделей и моделей машинного обучения.

ГЛАВА 1. ОСНОВЫ РАБОТЫ С ДАННЫМИ В PYTHON

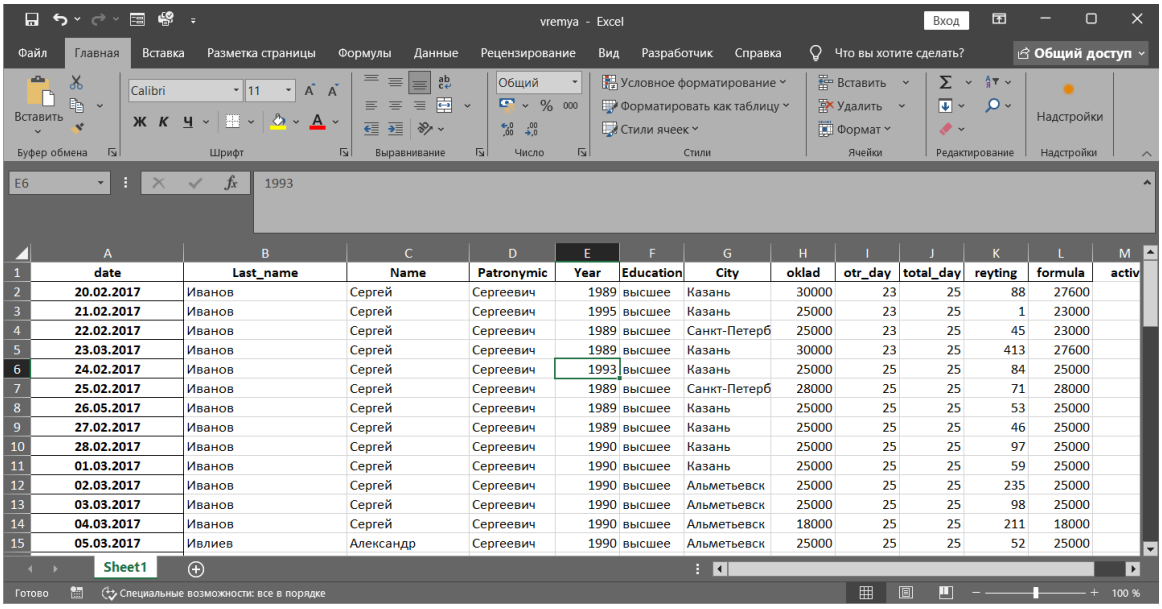
Обработка данных является критически важным первым этапом аналитической работы. В этой главе рассматриваются инструменты и методы на языке Python, необходимые для загрузки, очистки, преобразования и первичного анализа данных, с особым акцентом на работу с временными рядами. Освоение библиотеки «pandas» [1] — основа для применения более сложных методов машинного обучения и статистики.

1.1. Библиотека Pandas: загрузка, обработка и первичный анализ данных

С чем мы будем работать:

- Набор данных: чтение и загрузка данных в Excel формате
- Структура и сортировка данных по дате
- Поиск данных по времени
- Применение функции DataFrame.resample()
- Вывод данных временных рядов с помощью модуля matplotlib
- Проверка и тестирование программы

В качестве примера рассмотрим набор данных «Сотрудники предприятия». Пример структуры файла Excel представлен на Рис. 1:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date	Last_name	Name	Patronymic	Year	Education	City	oklad	otr_day	total_day	reyting	formula	activ
2	20.02.2017	Иванов	Сергей	Сергеевич	1989	высшее	Казань	30000	23	25	88	27600	
3	21.02.2017	Иванов	Сергей	Сергеевич	1995	высшее	Казань	25000	23	25	1	23000	
4	22.02.2017	Иванов	Сергей	Сергеевич	1989	высшее	Санкт-Петерб	25000	23	25	45	23000	
5	23.03.2017	Иванов	Сергей	Сергеевич	1989	высшее	Казань	30000	23	25	413	27600	
6	24.02.2017	Иванов	Сергей	Сергеевич	1993	высшее	Казань	25000	25	25	84	25000	
7	25.02.2017	Иванов	Сергей	Сергеевич	1989	высшее	Санкт-Петерб	28000	25	25	71	28000	
8	26.05.2017	Иванов	Сергей	Сергеевич	1989	высшее	Казань	25000	25	25	53	25000	
9	27.02.2017	Иванов	Сергей	Сергеевич	1989	высшее	Казань	25000	25	25	46	25000	
10	28.02.2017	Иванов	Сергей	Сергеевич	1990	высшее	Казань	25000	25	25	97	25000	
11	01.03.2017	Иванов	Сергей	Сергеевич	1990	высшее	Казань	25000	25	25	59	25000	
12	02.03.2017	Иванов	Сергей	Сергеевич	1990	высшее	Альметьевск	25000	25	25	235	25000	
13	03.03.2017	Иванов	Сергей	Сергеевич	1990	высшее	Альметьевск	25000	25	25	98	25000	
14	04.03.2017	Иванов	Сергей	Сергеевич	1990	высшее	Альметьевск	18000	25	25	211	18000	
15	05.03.2017	Ивлиев	Александр	Сергеевич	1990	высшее	Альметьевск	25000	25	25	52	25000	

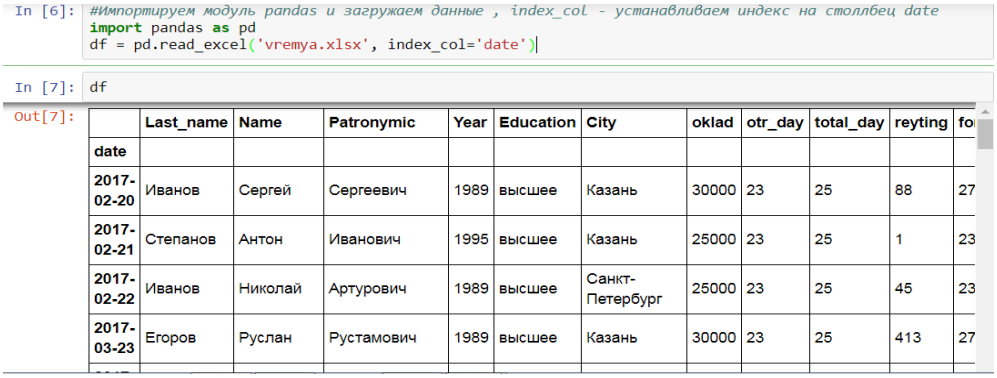
Рис. 1. Пример набора данных в файле

Практические шаги:

1. **Загрузка данных read.** Импортируем модуль pandas и загружаем данные из файла «vremya.xlsx». Параметр «index_col» устанавливает столбец «date» в качестве индекса DataFrame.

```python
import pandas as pd
df = pd.read_excel('vremya.xlsx', index_col='date')
```

В результате получаем DataFrame (Рис. 2), проиндексированный по дате, что является стандартом для работы с временными рядами.



In [6]:
```
#Импортируем модуль pandas и загружаем данные , index_col - устанавливаем индекс на столлбец date
import pandas as pd
df = pd.read_excel('vremya.xlsx', index_col='date')
```

In [7]:
```
df
```

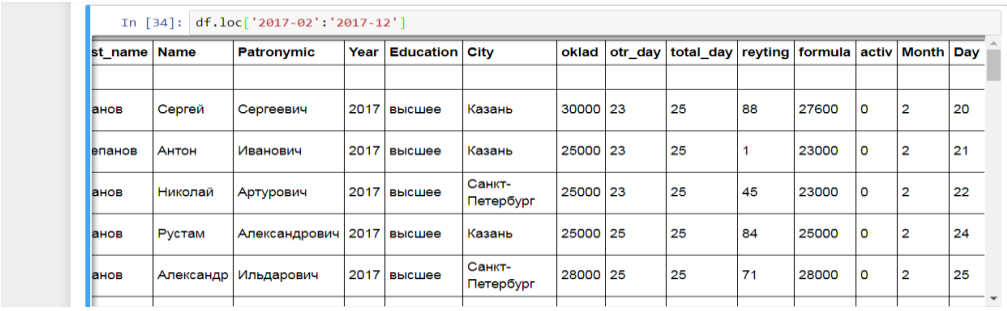
Out[7]:

	Last_name	Name	Patronymic	Year	Education	City	oklad	otr_day	total_day	reyting	fo
date											
2017-02-20	Иванов	Сергей	Сергеевич	1989	высшее	Казань	30000	23	25	88	27
2017-02-21	Степанов	Антон	Иванович	1995	высшее	Казань	25000	23	25	1	23
2017-02-22	Иванов	Николай	Артурович	1989	высшее	Санкт-Петербург	25000	23	25	45	23
2017-03-23	Егоров	Руслан	Рустамович	1989	высшее	Казань	30000	23	25	413	27

Рис. 2. Фрагмент полученного набора данных

2. **Индексация и выборка данных.** Выполним поиск данных по указанному периоду (например, с февраля 2017 по декабрь 2017 г.) (Рис. 3) с помощью локатора «.loc[]».

```python
df_period = df.loc['2017-02':'2017-12']
 print(df_period)
```



In [34]:
```
df.loc['2017-02':'2017-12']
```

st_name	Name	Patronymic	Year	Education	City	oklad	otr_day	total_day	reyting	formula	activ	Month	Day
анов	Сергей	Сергеевич	2017	высшее	Казань	30000	23	25	88	27600	0	2	20
епанов	Антон	Иванович	2017	высшее	Казань	25000	23	25	1	23000	0	2	21
анов	Николай	Артурович	2017	высшее	Санкт-Петербург	25000	23	25	45	23000	0	2	22
анов	Рустам	Александрович	2017	высшее	Казань	25000	25	25	84	25000	0	2	24
анов	Александр	Ильдарович	2017	высшее	Санкт-Петербург	28000	25	25	71	28000	0	2	25

Рис. 3. Результат выборки данных за период 2017 г.

Задание для самостоятельной работы: *Выведите информацию только за февраль 2017 г.*

1.2. Анализ временных рядов: индексирование, ресемплинг и визуализация

Временные ряды [2] требуют специальных методов анализа. Pandas предоставляет мощные инструменты для работы с временными индексами (DatetimeIndex), включая изменение частоты наблюдений (ресемплинг) и агрегацию данных.

Ресемплинг (Resampling)

Метод «resample()» объекта DataFrame разбивает DatetimeIndex на временные интервалы (периоды) и группирует данные по этим интервалам. Метод resample () возвращает объект Resampler, аналогичный объекту GroupBy, после чего можно применить агрегирующую функцию: mean(), min(), max() и т. д.

Частоты в pandas задаются специальными строковыми кодами:

- почасовой («H»)
- ежедневный («D»)
- еженедельный («W»)
- ежемесячный («M»)
- квартальный («Q»)
- годовой («A») и многие другие.

Пример. Вывод среднего выбранного значение по месяцам (“M”)

```python
# Выбираем столбцы для анализа
vremya = ['oklad', 'formula', 'activ']
# Применяем ресемплинг с агрегацией по среднему значению
sredn_znach = df[vremya].resample('M').mean()
print(sredn_znach)
```

```
In [43]: vremya = ['oklad','formula','activ']
         sredn_znach = df[vremya].resample('M').mean()
         sredn_znach
```

Out[43]:

	oklad	formula	activ
date			
2017-02-28	26142.857143	25228.571429	0.000000
2017-03-31	24733.333333	24653.333333	0.466667
2017-04-30	23750.000000	23750.000000	0.500000
2017-05-31	25000.000000	25000.000000	0.000000
2017-06-30	25550.000000	25550.000000	0.000000
2017-07-31	21000.000000	21000.000000	0.000000
2017-08-31	35000.000000	34400.000000	0.000000

Рис. 4. Средние значения показателей, агрегированные по месяцам

Задание для самостоятельной работы: Выполните вывод среднего значения по неделям.

Практикум

Получим информацию о количестве активных сотрудников по месяцам, используя агрегирующую функцию count() (Рис. 5).

```
In [44]: #Resampling мощный инструмент при работе с временными рядами (time series),
         #помогающий переформировать выборку так, как удобно вам.
         #Метод resample первым аргументом принимает строку rule.

         #Например, получим информацию о количестве сотрудников, которые активно проявили себя
         #на предприятие
         df.resample('M')['activ'].count()
```

```
Out[44]: date
         2017-02-28     7
         2017-03-31    30
         2017-04-30     4
         2017-05-31     1
         2017-06-30     1
         2017-07-31     1
```

Рис. 5. Количество активных сотрудников по месяцам

Визуализация временного ряда. Для визуализации данных удобно использовать библиотеку matplotlib (Рис. 6).

```python
import matplotlib.pyplot as plt
# Выбираем данные об активности за 2017 год
new_sample_df = df.loc['2017-02':'2017-12', ['activ']]
# Строим линейный график
new_sample_df.plot()
```

```
plt.title('Активность сотрудников (2017 год)')
plt.ylabel('Уровень активности')
plt.xlabel('Дата')
plt.grid(True)
                                    1plt.show()
```

```
#Берём активность сотрудников за 2017 год.
import matplotlib.pyplot as plt
new_sample_df = df.loc['2017-02':'2017-12', ['activ']]
new_sample_df.plot()
plt.show()
```

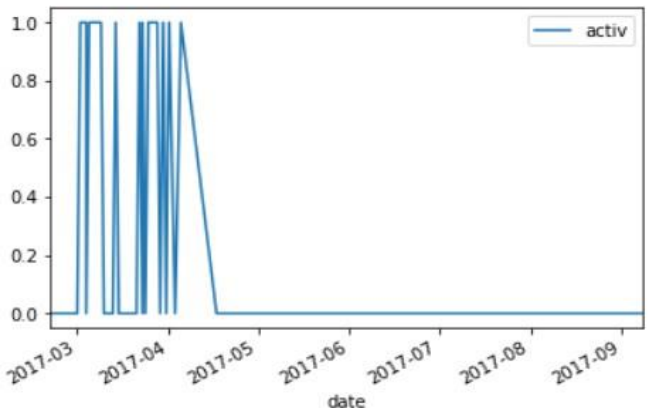


Рис. 6. График активности сотрудников за 2017 год

Задание к главе 1

- Создайте набор данных (согласно примеру Рис. 1) vremya.xlsx. Выполните считывание файла и преобразование набора данных в DataFrame, установив столбец date в качестве индекса. Убедитесь, что индекс имеет тип DatetimeIndex.
- Определите средний оклад на предприятии за период с февраля по декабрь 2017 года.
- Определите минимальный оклад на предприятии за февраль 2017 года.
- Используя метод resample, получите информацию о количестве сотрудников, проявивших активность, в разрезе: по месяцам; по неделям; по кварталам; за каждый год.
- Постройте график активности сотрудников за 2017 год с помощью matplotlib.

ГЛАВА 2. СТАТИСТИЧЕСКИЙ АНАЛИЗ И ПРОВЕРКА ГИПОТЕЗ

Прежде чем строить прогнозные модели, необходимо понять свойства данных. Для временных рядов ключевым понятием является стационарность. Эта глава знакомит с проверкой временных рядов на стационарность с помощью теста Дики-Фуллера [3-4] и основами регрессионного анализа для изучения взаимосвязей между переменными.

2.1. Введение в стационарность временных рядов

Стационарность [5] - свойство временного ряда, означающее, что его статистические характеристики (среднее значение, дисперсия, автоковариация) не меняются с течением времени. Большинство моделей прогнозирования (например, ARIMA) требуют стационарности ряда. Нестационарный ряд часто имеет тренд или сезонность.

2.2. Тест Дики-Фуллера: проверка на стационарность

Тест Дики-Фуллера (Augmented Dickey-Fuller test, ADF) — это статистический тест для проверки нулевой гипотезы о наличии единичного корня (нестационарности) во временном ряду.

- Нулевая гипотеза (H0): Ряд нестационарен (имеет единичный корень).
- Альтернативная гипотеза (H1): Ряд стационарен.
- Критерием принятия решения является p-value:
    - Если p-value < 0.05 (или другого уровня значимости), то отвергаем H0 и считаем ряд стационарным.
    - Если p-value >= 0.05, то ряд нестационарен.

2.3. Построение и интерпретация регрессионных моделей

Регрессионный анализ [6-7] позволяет моделировать и анализировать отношения между зависимой переменной и одной или несколькими независимыми переменными. Модель линейной регрессии предполагает, что зависимая переменная y линейно связана с параметрами модели

(коэффициентами β). В рамках данной главы мы рассмотрим простую линейную регрессию для выявления связи между двумя временными рядами.

Практикум

Выполним анализ стационарности и регрессии для временных рядов ВВП и цен. Перед выполнением практикума установите библиотеку statsmodels: **pip install statsmodels.** *Пример реализации:*

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
# Создание временного ряда
dates = pd.date_range(start='2019-01-01', periods=36, freq='M')
gdp = [1500, 1450, 1550, 1600, 1580, 1620, 1650, 1700, 1750, 1800, 1850, 1900,
       1950, 2000, 2050, 2100, 2150, 2200, 2250, 2300, 2350, 2400, 2450, 2500,
       2550, 2600, 2650, 2700, 2750, 2800, 2850, 2900, 2950, 3000, 3050, 3100]
prices = [100, 102, 104, 106, 108, 110, 112, 115, 118, 120, 125, 128,
          130, 132, 135, 138, 140, 145, 150, 155, 160, 165, 170, 175,
          180, 185, 190, 195, 200, 205, 210, 215, 220, 225, 230, 235]
# Создание DataFrame
data = pd.DataFrame({'Date': dates, 'GDP': gdp, 'Prices': prices})
data.set_index('Date', inplace=True)
# Проверка на стационарность (тест Дики-Фуллера)
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):
```

```
        result = adfuller(timeseries)
        print('ADF Statistic:', result[0])
        print('p-value:', result[1])
    print("Тест на стационарность для ВВП:")
    test_stationarity(data['GDP'])
    print("\nТест на стационарность для цен:")
    test_stationarity(data['Prices'])
    # Построение регрессионной модели
    X = sm.add_constant(data['GDP'])  # Добавление константы
    model = sm.OLS(data['Prices'], X).fit()
    # Результаты модели
    print("\nРезультаты регрессии:\n", model.summary())
    # Визуализация
    plt.figure(figsize=(12, 6))
    # График ВВП и цен
    plt.subplot(2, 1, 1)
    plt.plot(data.index,    data['GDP'],    label='ВВП    (трлн
рублей)', color='blue')
    plt.title('Временной ряд ВВП')
    plt.xlabel('Дата')
    plt.ylabel('ВВП')
    plt.legend()
    plt.subplot(2, 1, 2)
    plt.plot(data.index,     data['Prices'],     label='Цены',
color='orange')
    plt.title('Временной ряд цен')
    plt.xlabel('Дата')
    plt.ylabel('Цены')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

Интерпретация результатов:

1. Тест Дики-Фуллера: Проанализируйте p-value для рядов ВВП и цен. Если p-value > 0.05, ряд нестационарен. Часто экономические ряды, такие как ВВП, демонстрируют тренд и являются нестационарными.
2. Анализ регрессионной модели: При анализе регрессионной модели обратите внимание на R-squared (коэффициент детерминации), который показывает долю дисперсии зависимой переменной (цен), объясняемую моделью. Также важны coef для ВВП (наклон линии регрессии) и P>|t| для него же (p-value для коэффициента). Если P>|t| для ВВП меньше 0.05, связь между ВВП и ценами статистически значима.

ГЛАВА 3. МОДЕЛИРОВАНИЕ МНОГОМЕРНЫХ ЗАВИСИМОСТЕЙ

Традиционные меры зависимости, такие как корреляция Пирсона, часто оказываются недостаточными для описания сложных взаимосвязей в реальных данных. Эта глава посвящена теории копул [8-10] - инструменту для моделирования многомерных распределений, который позволяет раздельно описывать маргинальные распределения переменных и зависимость между ними.

3.1. Понятие о копулах

В современной аналитике - будь то финансы, метеорология, медицина или инженерия — исследователи регулярно сталкиваются с необходимостью анализа многомерных данных. Традиционные методы, такие как линейная корреляция Пирсона, адекватно описывают лишь простые линейные зависимости. Однако реальные данные часто демонстрируют сложные нелинейные паттерны, асимметрию и, что особенно важно, **хвостовую зависимость** — явление, когда экстремальные значения в одной переменной систематически связаны с экстремальными значениями в другой. Именно в условиях кризисов или аномальных событий такая зависимость проявляется наиболее сильно, что делает её учет критически важным для управления рисками.

Теория копул предлагает мощный и гибкий аппарат для решения этой задачи. **Копула** — это многомерная функция распределения, заданная на единичном гиперкубе, позволяющая «разделить» маргинальные распределения отдельных случайных величин и их совместную зависимость.

3.2. Гауссовы и t-копулы: анализ хвостовой зависимости

На практике широко используются два основных семейства эллиптических копул:

- Гауссова копула (нормальная копула): Строится на основе многомерного нормального распределения. Не имеет хвостовой зависимости, то есть экстремальные события предполагаются независимыми.
- t-копула (Копула Стьюдента): Основана на многомерном t-распределении. Ключевая особенность — наличие хвостовой зависимости, что означает повышенную вероятность совместного наступления экстремальных событий (как положительных, так и отрицательных).

Практикум

```python
# Python: фрагмент кода использования библиотеки copulae
# ... (Ваш код загрузки и подготовки данных)..
from copulae import GaussianCopula
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# 1. Выбор данных (столбцы с 4 по 10)
data = transposed_df.iloc[:, 3:10].values  # Индексация с 0, поэтому 3:10 - это столбцы 4-10
# 2. Создание и обучение гауссовой копулы
copula = GaussianCopula(dim=data.shape[1])
copula.fit(data)
# 3. Получение и визуализация матрицы корреляции
correlation_matrix = copula.sigma
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix,annot=True, cmap='coolwarm', fmt='.2f',
xticklabels=transposed_df.columns[3:10], yticklabels=transposed_df.columns[3:10])
plt.title('Матрица корреляции (зависимости) для столбцов 4-10')
plt.show()
```

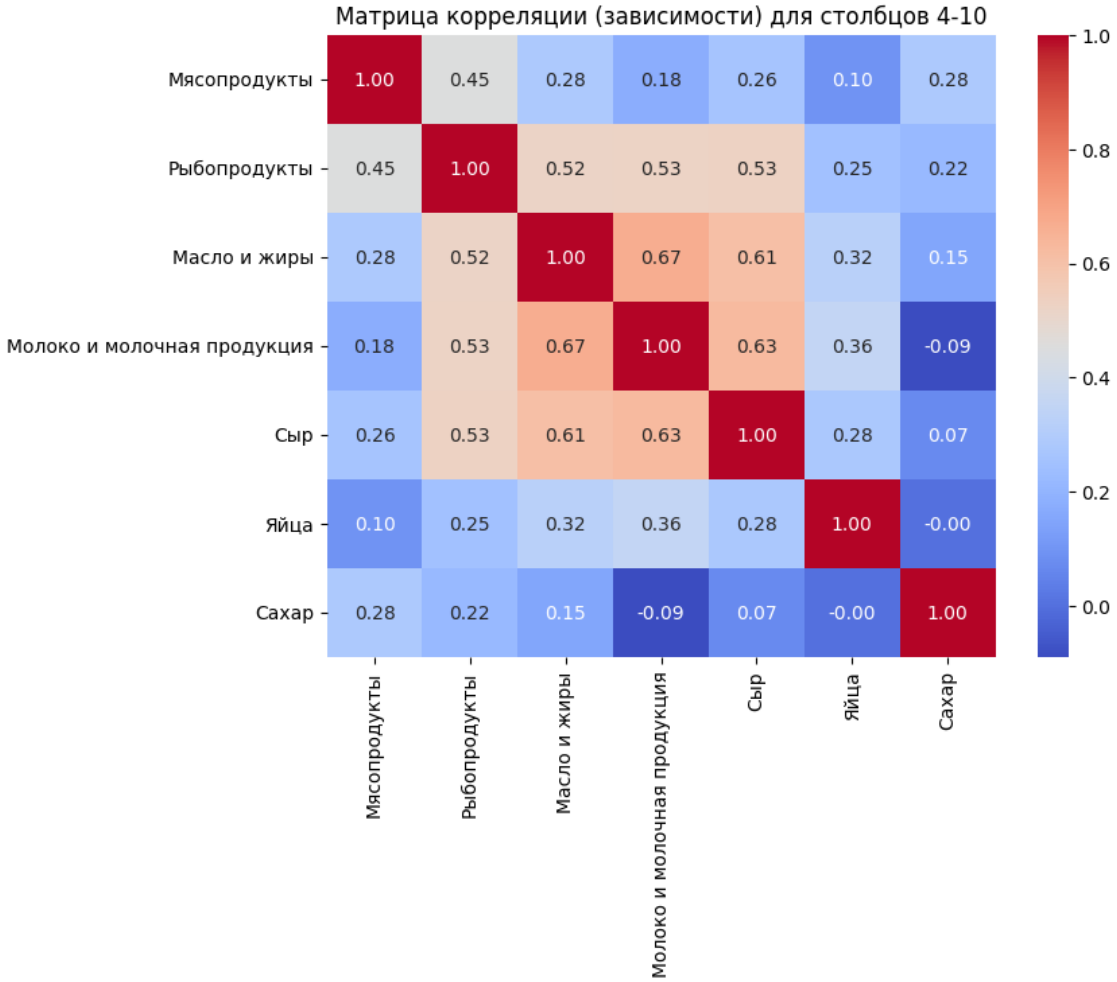


Рис. 7. Пример матрицы корреляции (зависимости)

Далее идет код для t-копулы и расчета хвостовой зависимости.

```python
from copulae import StudentCopula
import numpy as np
from scipy.stats import t
# 1. Выбор данных (столбцы с 4 по 10)
data = transposed_df.iloc[:, 3:10].values  # Индексация с 0, поэтому 3:10 — это столбцы 4-10
# 2. Создание копулы (например, t-копула для анализа хвостовой зависимости)
copula = StudentCopula(dim=data.shape[1])  # t-копула лучше подходит для хвостовой зависимости
# 3. Обучение копулы на данных
copula.fit(data)
```

```
# 4. Проверка структуры copula.params
print(copula.params)  # Посмотрим, как выглядят параметры
# 5. Извлечение параметров
df = copula.params.df  # степени свободы
rho = copula.params.rho  # одномерный массив корреляций
# 6. Выбор корреляции между двумя переменными (например, первая и вторая переменная)
# Для многомерной копулы нужно выбрать две переменные и их корреляцию
# Например, корреляция между первой и второй переменной:
rho_12 = rho[0]  # Предположим, что rho[0] — это корреляция между первой и второй переменной
# 7. Ручное вычисление хвостовой зависимости
def tail_dependence_t_copula(df, rho):
    nu = df  # степени свободы
    upper_tail = 2 * t.cdf(-np.sqrt((nu + 1) * (1 - rho) / (1 + rho)), nu + 1)
    lower_tail = upper_tail  # для t-копулы верхняя и нижняя хвостовая зависимость одинаковы
    return lower_tail, upper_tail
lower_tail, upper_tail = tail_dependence_t_copula(df, rho_12)
# 8. Вывод результатов
print("Нижняя хвостовая зависимость:")
print(lower_tail)
print("\nВерхняя хвостовая зависимость:")
print(upper_tail)
for i, rho_ij in enumerate(rho):
lower_tail, upper_tail = tail_dependence_t_copula(df, rho_ij)
print(f"Пара переменных {i+1}: Нижняя хвостовая зависимость = {lower_tail}, Верхняя хвостовая зависимость = {upper_tail}")
```

```
StudentParams(df=21.022346199730567, rho=array([ 0.43331562,  0.28864292,  0.17275095,  0.26489759,  0.10483036,
        0.26232587,  0.51389948,  0.5385835 ,  0.53185985,  0.23770976,
        0.2016989 ,  0.67671867,  0.61776284,  0.31523869,  0.14231322,
        0.64023523,  0.34523335, -0.09563634,  0.27248398,  0.04768939,
       -0.00547868]))
Нижняя хвостовая зависимость:
0.00738498160741122

Верхняя хвостовая зависимость:
0.00738498160741122
Пара переменных 1: Нижняя хвостовая зависимость = 0.00738498160741122, Верхняя хвостовая зависимость = 0.00738498160741122
Пара переменных 2: Нижняя хвостовая зависимость = 0.0020875926488417187, Верхняя хвостовая зависимость = 0.0020875926488417187
Пара переменных 3: Нижняя хвостовая зависимость = 0.0006946778404311168, Верхняя хвостовая зависимость = 0.0006946778404311168
Пара переменных 4: Нижняя хвостовая зависимость = 0.0016781644465373548, Верхняя хвостовая зависимость = 0.0016781644465373548
Пара переменных 5: Нижняя хвостовая зависимость = 0.0003483673215915644, Верхняя хвостовая зависимость = 0.0003483673215915644
Пара переменных 6: Нижняя хвостовая зависимость = 0.001638610892705484, Верхняя хвостовая зависимость = 0.001638610892705484
Пара переменных 7: Нижняя хвостовая зависимость = 0.014322778137366463, Верхняя хвостовая зависимость = 0.014322778137366463
Пара переменных 8: Нижняя хвостовая зависимость = 0.01746164510425787, Верхняя хвостовая зависимость = 0.01746164510425787
Пара переменных 9: Нижняя хвостовая зависимость = 0.016547419046277156, Верхняя хвостовая зависимость = 0.016547419046277156
Пара переменных 10: Нижняя хвостовая зависимость = 0.0013013716216040009, Верхняя хвостовая зависимость = 0.0013013716216040009
Пара переменных 11: Нижняя хвостовая зависимость = 0.0009222791295928492, Верхняя хвостовая зависимость = 0.0009222791295928492
Пара переменных 12: Нижняя хвостовая зависимость = 0.05134794299922689, Верхняя хвостовая зависимость = 0.05134794299922689
Пара переменных 13: Нижняя хвостовая зависимость = 0.032573538306620145, Верхняя хвостовая зависимость = 0.032573538306620145
Пара переменных 14: Нижняя хвостовая зависимость = 0.002655314653706485, Верхняя хвостовая зависимость = 0.002655314653706485
Пара переменных 15: Нижняя хвостовая зависимость = 0.0005121923936205 41, Верхняя хвостовая зависимость = 0.0005121923936205 41
Пара переменных 16: Нижняя хвостовая зависимость = 0.03877201000850041, Верхняя хвостовая зависимость = 0.03877201000850041
Пара переменных 17: Нижняя хвостовая зависимость = 0.0034663303216290204, Верхняя хвостовая зависимость = 0.0034663303216290204
Пара переменных 18: Нижняя хвостовая зависимость = 3.523348450069917e-05, Верхняя хвостовая зависимость = 3.523348450069917e-05
Пара переменных 19: Нижняя хвостовая зависимость = 0.0018000643374273505, Верхняя хвостовая зависимость = 0.0018000643374273505
Пара переменных 20: Нижняя хвостовая зависимость = 0.00018908559038780857, Верхняя хвостовая зависимость = 0.00018908559038780857
Пара переменных 21: Нижняя хвостовая зависимость = 0.00010411151741603273, Верхняя хвостовая зависимость = 0.00010411151741603273
```

Рис. 8. Фрагмент результата анализа в хвостах

Практикум

Сравнительный анализ зависимостей

Используя приведенный выше код, выполните следующие действия:

1. Постройте матрицы корреляции как для Гауссовой, так и для t-копулы. Сравните их визуально.
2. Проанализируйте рассчитанные значения хвостовой зависимости для различных пар переменных. Какие пары демонстрируют наибольшую и наименьшую хвостовую зависимость?
3. Сделайте вывод о том, какая из копул (Гауссова или t-копула) лучше подходит для описания зависимостей в ваших данных, исходя из наличия хвостовой зависимости.

ГЛАВА 4. АНСАМБЛИ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

Ансамбли методов машинного обучения позволяют объединять несколько моделей для создания одной, более точной и устойчивой. Градиентный бустинг [11] - один из наиболее мощных алгоритмов в этом семействе методов (ансамблевых методов). Данная глава раскрывает теорию градиентного бустинга, знакомит с популярной библиотекой и предлагает реализацию алгоритма "вручную" для глубокого понимания его работы.

4.1. Теория градиентного бустинга

Принцип последовательного исправления ошибок

XGBoost (Extreme Gradient Boosting) — это высокопроизводительная и масштабируемая реализация градиентного бустинга, отличающаяся скоростью, параллельной обработкой, встроенной регуляризацией и способностью эффективно работать с большими наборами данных.

Основная идея бустинга заключается в последовательном построении ансамбля "слабых" моделей (например, неглубоких деревьев), где каждая следующая модель обучается предсказывавать ошибки (остатки) ансамбля, построенного на предыдущем шаге. Градиентный бустинг обобщает этот подход, используя градиентный спуск в пространстве функций для минимизации ошибки.

4.2. Алгоритм XGBoost: основные гиперпараметры и их настройка

Гиперпараметры

- Количество деревьев (n_estimators). Сколько раз мы будем корректировать ошибки
- Скорость обучения (learning_rate). Насколько сильно корректируем при каждом шаге
- Глубина деревьев (max_depth)

Помимо n_estimators, learning_rate и max_depth, ключевыми гиперпараметрами являются:

- reg_alpha и reg_lambda: L1 и L2 регуляризация для предотвращения переобучения.
- subsample: Доля наблюдений, используемых для обучения каждого дерева.
- colsample_bytree: Доля признаков, используемых для построения каждого дерева.

4.3. Пример реализация градиентного бустинга на Python с помощью библиотеки Scikit-learn

Пример с квартирами (визуализация):

df

	район	площадь	стоимость	район_код
0	центр	50	3000000	1
1	центр	70	3500000	1
2	пригород	30	1500000	0
3	пригород	60	2000000	0

Next steps: New interactive sheet

Рис. 9. Фрагмент набора данных

```python
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
# Создаем пример данных
data = {
    'район': ['центр', 'центр', 'пригород', 'пригород'],
    'площадь': [50, 70, 30, 60],
    'стоимость': [3000000, 3500000, 1500000, 2000000]
}
df = pd.DataFrame(data)
print("=== ИСХОДНЫЕ ДАННЫЕ ===")
```

```
print(df)
# Преобразуем категориальные переменные
df['район_код'] = df['район'].map({'центр': 1, 'пригород': 0})
# Признаки и целевая переменная
X = df[['район_код', 'площадь']]
y = df['стоимость']
```

Процесс обучения шаг за шагом

```
Шаг 0: Начальное предположение
Все квартиры стоят 2,500,000 ₽ (среднее значение)
Ошибка: Большая!
Шаг 1: Первое дерево
if район == 'центр':
   добавить 200,000 ₽
else:
   вычесть 200,000 ₽
Шаг 2: Второе дерево (учится на ошибках)
if площадь < 40:
    вычесть 500,000 ₽
Шаг 3: Третье дерево (уточнение)
if район == 'центр' and площадь >= 70:
    добавить 500,000 ₽
```

Практикум

Подготовка среды для разработки программного модуля:

```
# Установка необходимых библиотек
!pip install pandas numpy matplotlib scikit-learn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
Шаг 1: Создаем данные
```

```
 # Данные о квартирах
 data = {
     'район': ['центр', 'центр', 'Пригород', 'Пригород'],
     'площадь': [50, 70, 30, 60],
     'стоимость': [3000000, 3500000, 1500000, 2000000]
 }df = pd.DataFrame(data)
 print("Наши данные:")
 print(df)
 Шаг 2: Преобразуем данные
 # Машинное обучение понимает только числа
 df['район_код'] = df['район'].map({'центр': 1, 'Пригород':
0})
 # Разделяем на признаки и целевую переменную
 X = df[['район_код', 'площадь']]
 y = df['стоимость']
 print("\nПризнаки (X):")
 print(X)
 print("\nЦелевая переменная (y):")
 print(y)
 Шаг 3: Базовая модель (1 дерево)
 # Обучаем простое дерево
 simple_tree = DecisionTreeRegressor(max_depth=2,
random_state=42)
 simple_tree.fit(X, y)
 # Предсказания
 simple_predictions = simple_tree.predict(X)
 print("\nПредсказания простого дерева:")
 for i, (real, pred) in enumerate(zip(y,
simple_predictions)):
     print(f"Квартира {i+1}: Реально {real:,} → Предсказано
{pred:,.0f}")
 Шаг 4: Градиентный бустинг (вручную)
 # Параметры
```

```
n_estimators = 10
learning_rate = 0.1
max_depth = 2
# Инициализация
predictions = np.full(len(y), y.mean())
print(f"\nНачальное предсказание: {y.mean():,.0f} руб.")
models = []
residuals_history = []
for i in range(n_estimators):
    # 1. Вычисляем ошибки (остатки)
    residuals = y - predictions
    # 2. Обучаем дерево на ошибках
    tree = DecisionTreeRegressor(max_depth=max_depth,
random_state=42)
    tree.fit(X, residuals)
    # 3. Предсказания текущего дерева
    tree_pred = tree.predict(X)
    # 4. Обновляем общие предсказания
    predictions += learning_rate * tree_pred
    # Сохраняем для анализа
    models.append(tree)
    residuals_history.append(residuals.copy())
    print(f"\n--- Шаг {i+1} ---")
    print(f"Средняя ошибка:
{np.mean(np.abs(residuals)):,.0f} руб.")
Шаг 5: Визуализация процесса
# График обучения
plt.figure(figsize=(15, 5))
# 1. Снижение ошибки
plt.subplot(1, 3, 1)
errors = [np.mean(np.abs(res)) for res in
residuals_history]
plt.plot(errors, marker='o')
```

```
plt.title('Снижение ошибки')
plt.xlabel('Шаг обучения')
plt.ylabel('Средняя ошибка (руб)')
plt.grid(True)
# 2. Сравнение реальных и предсказанных значений
plt.subplot(1, 3, 2)
plt.scatter(y, predictions, alpha=0.7)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.title('Реальные vs Предсказанные')
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.grid(True)
# 3. Финальные ошибки
plt.subplot(1, 3, 3)
final_errors = y - predictions
plt.bar(range(len(final_errors)), final_errors)
plt.title('Финальные ошибки')
plt.xlabel('Номер квартиры')
plt.ylabel('Ошибка (руб)')
plt.grid(True)
plt.tight_layout()
plt.show()
Шаг 6: Предсказание для новых данных
# Новые квартиры для предсказания
new_apartments = pd.DataFrame({
    'район': ['центр', 'Пригород', 'центр'],
    'площадь': [65, 35, 80]
})
new_apartments['район_код'] =
new_apartments['район'].map({'центр': 1, 'Пригород': 0})
X_new = new_apartments[['район_код', 'площадь']]
```

```
 # Делаем предсказания
 final_predictions = np.full(len(X_new), y.mean())
 for tree in models:
     final_predictions += learning_rate *
tree.predict(X_new)
 new_apartments['предсказанная_стоимость'] =
final_predictions.astype(int)
 print("\nПредсказания для новых квартир:")
 print(new_apartments)
```

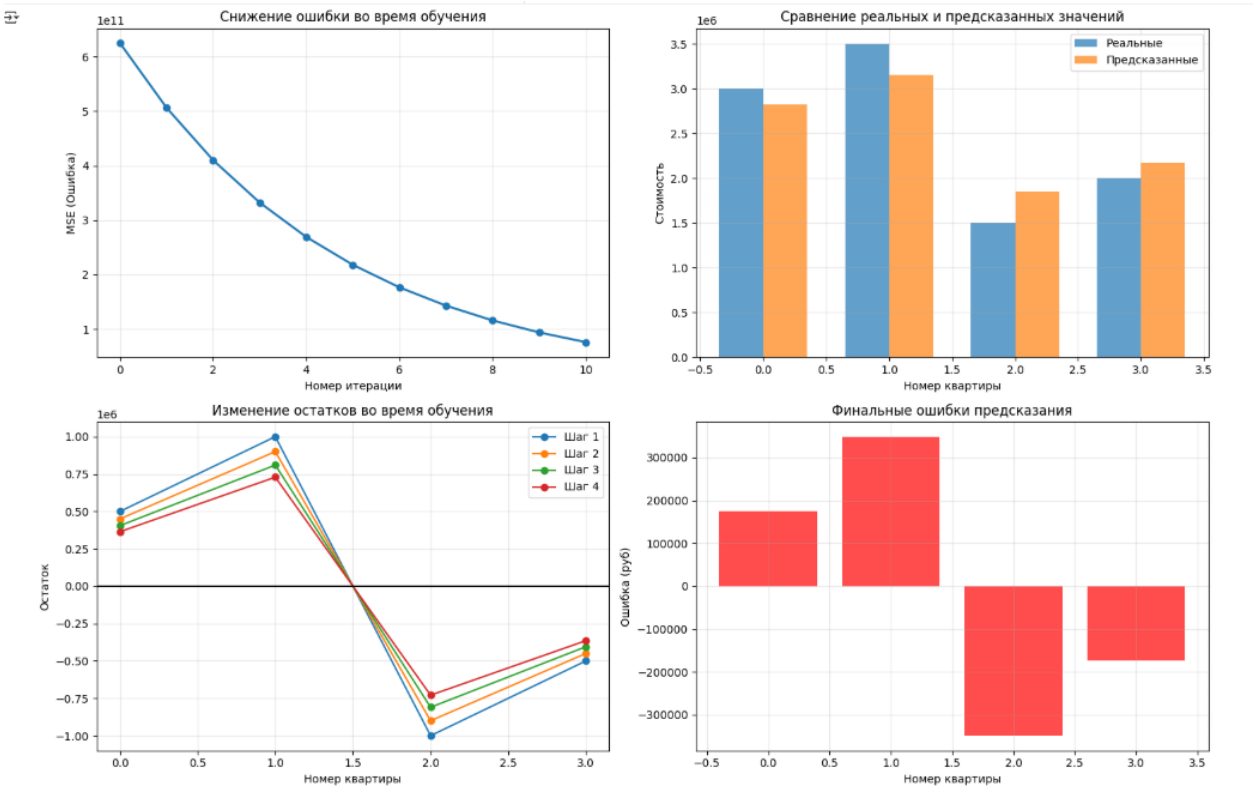


Рис. 10. Результат прогнозированиям (визуализация)

Результат вывода для новых квартир:

```
=== ПРЕДСКАЗАНИЯ ДЛЯ НОВЫХ КВАРТИР ===
      район  площадь  район_код  предсказанная_стоимость
0     центр        65          1                  3151321
1  пригород        35          0                  1848678
2     центр        80          1                  3151321
3  пригород        45          0                  1848678
```

```
=== АНАЛИЗ МОДЕЛИ ===
Важность признаков (последнее дерево):
район_код: 0.900
площадь: 0.100
```

Задание

Задание 1: Прогноз цен для нового ЖК

```
# Задача: Постройте прогноз для нового жилого комплекса
new_buildings = pd.DataFrame({
    'район': ['центр', 'центр', 'Пригород', 'Пригород'],
    'площадь': [45, 85, 55, 75],
    'тип_отделки': ['черновая', 'чистовая', 'черновая', 'чистовая']
})
# Как учесть тип отделки в модели?
```

Задание 2: Оптимизация портфеля недвижимости

Риелторская компания хочет оптимизировать портфель.

Задача: Определите, квартиры в каких районах и с какой площадью имеют наибольшую потенциальную доходность (разница между рыночной и прогнозируемой стоимостью)

Задание 3: Прогноз стоимости акций

```
# Адаптируйте модель для временных рядов акций:
stock_data = {
    'цена_открытия': [100, 102, 101, 105, 108],
    'объем_торгов': [1000000, 1200000, 800000, 1500000, 900000],
    'волатильность': [0.02, 0.015, 0.025, 0.018, 0.022],
    'цена_закрытия': [102, 101, 105, 108, 110]  # целевая переменная
}
```

Задание 4: Оценка риска страхования

```
# Страховая компания: прогноз стоимости страховых выплат
insurance_data = {
    'возраст_водителя': [25, 45, 35, 28, 50],
    'мощность_авто': [150, 200, 120, 180, 250],
    'стаж_водителя': [3, 20, 8, 5, 25],
    'стоимость_выплат': [50000, 20000, 30000, 45000, 15000]
}
```

Задание 5: Прогноз продаж магазина

```
# Сеть магазинов хочет прогнозировать ежедневные продажи
retail_data = {
    'день_недели': [1, 2, 3, 4, 5, 6, 7],  # 1-понедельник, 7-воскресенье
```

```
    'праздник': [0, 0, 1, 0, 0, 0, 1],  # 1 = праздничный день
    'погода': [1, 2, 1, 3, 2, 1, 2],  # 1-солнечно, 2-облачно, 3-дождь
    'продажи': [100000, 80000, 150000, 90000, 95000, 120000, 180000]
}
```

Задание 6: Прогноз длительности госпитализации

```
# Больница хочет оптимизировать койко-места
medical_data = {
    'возраст': [45, 65, 35, 28, 72],
    'тяжесть_заболевания': [2, 3, 1, 2, 4],  # шкала 1-5
    'количество_процедур': [3, 5, 2, 4, 6],
    'длительность_госпитализации': [7, 14, 5, 9, 21]  # дни
}
```

Задание 7: Диагностика заболеваний

```
# Модель для поддержки врачебных решений
diagnosis_data = {
    'уровень_сахара': [120, 180, 90, 150, 200],
    'давление': [130, 150, 120, 140, 160],
    'холестерин': [200, 250, 180, 220, 280],
    'диабет': [0, 1, 0, 1, 1]  # 1 = диагностирован диабет
}
```

Задание 8: Прогноз успеваемости студентов

```
# Университет хочет прогнозировать успеваемость
education_data = {
    'часы_обучения': [20, 35, 15, 25, 30],
    'посещаемость': [0.8, 0.9, 0.7, 0.85, 0.95],
    'баллы_вступительных': [75, 85, 65, 80, 90],
    'итоговый_балл': [3.5, 4.2, 3.0, 3.8, 4.5]
}
```

Библиографический список

1. Титов, А. Н. Обработка данных в Python. Основы работы с библиотекой Pandas : Учебно-методическое пособие / А. Н. Титов, Р. Ф. Тазиева. – Казань : Казанский национальный исследовательский технологический университет, 2022. – 116 с. – ISBN 978-5-7882-3164-8.
2. Газизов, Д. И. Обзор методов статистического анализа временных рядов и проблемы, возникающие при анализе нестационарных временных рядов / Д. И. Газизов // Научный журнал. – 2016. – № 3(4). – С. 9-14.
3. Смирнова, И. В. Определение порядка интегрируемости экономических временных рядов / И. В. Смирнова, Л. Н. Клянина // Инженерный вестник Дона. – 2016. – № 2(41). – С. 42.
4. Гребенюк, Е. А. Применение методов эконометрического анализа данных для идентификации и датирования "пузырей" на финансовых рынках / Е. А. Гребенюк, А. В. Малинкина // Проблемы управления. – 2014. – № 5. – С. 50-58.
5. Евграфов, А. В. Оценка стационарности рядов метеоданных при динамическом моделировании стока / А. В. Евграфов, И. М. Евграфова // Природообустройство. – 2019. – № 4. – С. 78-81. – DOI 10.34677/1997-6011/2019-4-78-82.
6. Технология анализа, прогноза и оптимизации работы группы скважин с помощью регрессионного анализа и характеристик вытеснения / И. В. Афанаскин, А. А. Колеватов, М. Ю. Ахапкин [и др.] // Геология, геофизика и разработка нефтяных и газовых месторождений. – 2022. – № 11(371). – С. 60-70. – DOI 10.33285/2413-5011-2022-11(371)-60-70.
7. Борисова, Л. Р. Анализ продолжительности жизни в регионах Российской Федерации по методам машинного обучения и регрессионному анализу / Л. Р. Борисова // Современная математика и концепции инновационного математического образования. – 2021. – Т. 8, № 1. – С. 224-230.
8. Столярова, В. Ф. Копулы и моделирование зависимости: косвенные оценки интенсивности рискованного поведения / В. Ф. Столярова // Компьютерные

инструменты в образовании. – 2018. – № 3. – С. 22-37. – DOI 10.32603/2071-2340-3-22-37.

9. К вопросу оценки надежности нанокомпонентов с использованием понятия копулы / О. Ю. Кондратьева, Д. В. Терин, Р. А. Сафонов [и др.] // Взаимодействие сверхвысокочастотного, терагерцового и оптического излучения с полупроводниковыми микро- и наноструктурами, метаматериалами и биообъектами : Материалы Всероссийской научной школы-семинара , Саратов, 14–15 мая 2015 года / Ответственный редактор Д.А. Усанов. – Саратов: Издательство "Саратовский источник", 2015. – С. 72-74.
10. Бронштейн, Е. М. Копулы специального вида и их применение при анализе состояния финансового рынка / Е. М. Бронштейн, А. Р. Зинурова // Прикладная эконометрика. – 2012. – № 3(27). – С. 109-114.
11. Gradient boosting method application to support process decisions in the electron-beam welding process / V. S. Tynchenko, I. A. Golovenok, V. E. Petrenko [et al.] // Siberian Journal of Science and Technology. – 2020. – Vol. 21, No. 2. – P. 206-214. – DOI 10.31772/2587-6066-2020-21-2-206-214.

Учебное издание

Бурнашев Рустам Арифович

ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ И МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОСТРОЕНИЯ УСТОЙЧИВЫХ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Учебно-методическое пособие

Подписано в печать 25.09.2025.
Бумага офсетная. Печать цифровая.
Формат 60x84 1/16. Гарнитура «Times New Roman».
Усл. печ. л. 0,7. Тираж 100 экз. Заказ 51/11.

Отпечатано с готового оригинал-макета
в типографии Издательства Казанского университета

420008, г. Казань, ул. Профессора Нужина, 1/37
тел. (843) 206-52-14 (1704), 206-52-14 (1705)