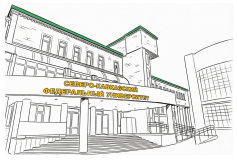

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Методические указания
к выполнению лабораторных работ

Направление подготовки: 09.03.02 «Информационные системы и технологии»
Профиль: «Прикладное программирование и интернет-технологии»

Квалификация (степень) выпускника: бакалавр
Форма обучения: очная

Ставрополь – 2026

УДК 004.65:004.42
ББК 32.973.2-018.2

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Методические указания к лабораторным работам

Аннотация: Методические указания содержат комплект из 8 лабораторных работ по дисциплине «Технологии обработки информации». Каждая работа включает цель, задачи, теоретический минимум, пошаговый разбор примера, фрагменты программного кода для решения прикладных задач, индивидуальное задание для самостоятельного выполнения, а также вопросы для самоконтроля. Материал охватывает полный цикл работы с данными: от проектирования информационных моделей и реляционных баз данных до веб-скрапинга, предобработки, разведочного анализа, визуализации, машинного обучения и обеспечения безопасности. Рекомендуемый объем — 32 часа лабораторных занятий. Работы выполняются в среде Python с использованием современных библиотек анализа данных, SQL-интерпретаторов и BI-инструментов.
Составитель: к.т.н. Д.Л. Винокурский
Рецензент: к.ф.м.н. Е.В. Крахоткина

Содержание

Введение

Дисциплина «Технологии обработки информации» занимает центральное место в подготовке специалистов, способных проектировать, реализовывать и сопровождать информационные системы, работающие с данными в условиях современных цифровых экосистем. Понимание полного жизненного цикла данных — от их генерации и структурирования до анализа, визуализации и защиты — является необходимым условием для принятия обоснованных решений и автоматизации бизнес-процессов.

Актуальность дисциплины обусловлена следующими факторами:

- Экспоненциальный рост объёмов генерируемых данных требует системных подходов к их структурированию и обработке;
- Переход к data-driven управлению делает навыки анализа и интерпретации данных обязательными для ИТ-специалистов;
- Развитие облачных платформ, BI-систем и средств машинного обучения меняет архитектуру информационных потоков;
- Ужесточение требований к защите персональных и коммерческих данных повышает значимость аспектов безопасности и аудита.

Цель методических указаний — предоставить обучающимся структурированный практический материал для освоения современных технологий обработки информации на примере сквозного цикла работы с данными.

Задачи лабораторного практикума:

1. Сформировать навыки проектирования информационных моделей и нормализации структур хранения;
2. Освоить язык SQL для манипулирования и анализа реляционных данных;
3. Научиться автоматизировать сбор данных из внешних источников и приводить их к структурированному виду;
4. Развить компетенции в предобработке, очистке и трансформации реальных датасетов;
5. Научиться проводить разведочный анализ, выявлять закономерности и проверять статистические гипотезы;
6. Освоить создание интерактивных визуализаций и дашбордов для поддержки принятия решений;
7. Получить практический опыт построения и валидации базовых моделей машинного обучения;
8. Изучить и применить методы обеспечения безопасности, аудита и соответствия нормативным требованиям.

Организация выполнения работ:

- Каждая лабораторная работа рассчитана на 2 академических часа;
- Перед началом работы студент обязан изучить теоретический материал и ответить на допусковые вопросы;
- Экспериментальная часть выполняется в локальной среде разработки или облачном Jupyter-окружении;
- Отчёт по работе должен содержать: цель, описание предметной области/датасета, листинги кода, результаты выполнения (графики, таблицы, метрики), выводы и ответы на контрольные вопросы;
- Защита работы включает демонстрацию работающего кода и устный ответ по теоретическим основам.

Требования к программному обеспечению:

- Язык программирования: Python 3.9+;
- Библиотеки: pandas, numpy, matplotlib, seaborn, scipy, scikit-learn, sqlite3, requests, beautifulsoup4, plotly, streamlit, hashlib, logging;
- Среда разработки: Jupyter Notebook, VS Code или PyCharm;
- Дополнительно: SQLiteBrowser, Git, Docker (по необходимости).

1 Проектирование информационной модели предметной области

Цель: Сформировать навыки проектирования концептуальных и логических информационных моделей, освоить методы нормализации реляционных структур.
Задачи:

- Изучить принципы ER-моделирования, сущности, атрибуты и типы связей;
- Построить концептуальную модель заданной предметной области;
- Выполнить преобразование в логическую модель и проверить соответствие нормальным формам (1NF–3NF);
- Реализовать схему в SQLite и провалидировать целостность связей.

Теоретическое описание: Информационная модель — абстрактное представление структуры данных предметной области. Концептуальная модель (ER-диаграмма) фокусируется на бизнес-сущностях и отношениях без привязки к СУБД. Логическая модель детализирует типы данных, ключи и ограничения. Нормализация устраняет избыточность и аномалии обновления: 1NF исключает повторяющиеся группы, 2NF устраняет частичные зависимости от составного ключа, 3NF удаляет транзитивные зависимости.
Разобранный пример: Предметная область: «Библиотечная система». Сущности: Книги, Авторы, Читатели, Выдачи. Связи: Автор-Книга (M:N), Книга-Выдача (1:N). Результат: 5 нормализованных таблиц с первичными/внешними ключами.
Программный код (генерация схемы в SQLite):

```
import sqlite3
conn = sqlite3.connect('library.db')
cursor = conn.cursor()
cursor.executescript("""
CREATE TABLE IF NOT EXISTS Authors (author_id INTEGER PRIMARY KEY
   AUTOINCREMENT, name TEXT NOT NULL);
CREATE TABLE IF NOT EXISTS Books (book_id INTEGER PRIMARY KEY
   AUTOINCREMENT, title TEXT NOT NULL, isbn TEXT UNIQUE, publish_year
    INTEGER);
CREATE TABLE IF NOT EXISTS AuthorBook (author_id INTEGER, book_id
   INTEGER, PRIMARY KEY (author_id, book_id), FOREIGN KEY (author_id)
    REFERENCES Authors(author_id), FOREIGN KEY (book_id) REFERENCES
   Books(book_id));
CREATE TABLE IF NOT EXISTS Readers (reader_id INTEGER PRIMARY KEY
   AUTOINCREMENT, full_name TEXT NOT NULL, email TEXT UNIQUE);
CREATE TABLE IF NOT EXISTS Loans (loan_id INTEGER PRIMARY KEY
   AUTOINCREMENT, reader_id INTEGER, book_id INTEGER, loan_date TEXT,
    return_date TEXT, FOREIGN KEY (reader_id) REFERENCES Readers(
   reader_id), FOREIGN KEY (book_id) REFERENCES Books(book_id));
""")
print("                                          .")
conn.close()
```

Индивидуальное задание: Спроектируйте модель для системы «Интернет-магазин» (Пользователи, Товары, Категории, Заказы, Платежи). Реализуйте схему в SQLite и добавьте 3–4 ограничения (CHECK, UNIQUE, DEFAULT).
Вопросы для самопроверки: 1) В чём разница между концептуальной и логической моделью? 2) Какие аномалии возникают при нарушении 2NF и 3NF? 3) Когда целесообразно использовать денормализацию?

2 Организация и манипулирование данными в реляционных СУБД

Цель: Освоить язык SQL для создания, наполнения и аналитической обработки реляционных данных.
Задачи:

- Изучить базовые конструкции DDL, DML и DQL;
- Выполнить сложные выборки с использованием JOIN, GROUP BY, подзапросов и оконных функций;
- Проанализировать производительность запросов и роль индексов;
- Реализовать транзакции для обеспечения целостности операций.

Теоретическое описание: SQL — стандартный язык взаимодействия с реляционными СУБД. Оконные функции (`OVER()`, `PARTITION BY`) позволяют вычислять агрегаты без свёртки строк. Индексы ускоряют поиск, но замедляют вставку. Транзакции гарантируют ACID-свойства.
Разобранный пример: Датасет: продажи по филиалам. Задача: найти топ-3 товара по выручке в каждом филиале за 2025 год. Решение: `ROW_NUMBER() OVER(PARTITION BY branch_id ORDER BY revenue DESC)`.
Программный код (аналитический запрос):

```
import sqlite3, pandas as pd
conn = sqlite3.connect('library.db')
conn.execute("INSERT INTO Authors (name) VALUES ('            '), ('
              ')")
conn.execute("INSERT INTO Books (title, isbn, publish_year) VALUES ('
   DataScience 101','111-1',2023)")
conn.commit()
query = """SELECT b.title, COUNT(l.loan_id) as loans, COUNT(l.loan_id
   ) OVER(PARTITION BY strftime('%Y', l.loan_date)) as yearly_loans
   FROM Books b JOIN Loans l ON b.book_id = l.book_id WHERE l.
   loan_date IS NOT NULL ORDER BY loans DESC;"""
df = pd.read_sql(query, conn)
print(df)
conn.close()
```

Индивидуальное задание: Загрузите датасет `employees`. Напишите запрос, вычисляющий среднюю зарплату по департаментам, ранжирующий сотрудников внутри отдела и отбирающий тех, чья зарплата выше медианной.
Вопросы для самопроверки: 1) Отличие INNER JOIN от LEFT JOIN? 2) Как индекс B-Tree ускоряет поиск? 3) Что гарантирует свойство изоляции транзакций?

3 Автоматизация сбора и первичной обработки данных (парсинг, импорт)

Цель: Научиться автоматизировать сбор данных из веб-источников и API, преобразовывать разнородные форматы в структурированный вид.
Задачи:

- Изучить протокол HTTP, REST-API и принципы парсинга HTML/JSON;
- Реализовать скрипт загрузки данных с обработкой ошибок и задержек;
- Преобразовать сырые данные в табличный формат;
- Сохранить результат в CSV/SQLite.

Теоретическое описание: Автоматизированный сбор основан на HTTP-запросах. JSON — стандарт обмена данными в API. Важно соблюдать `robots.txt`, ограничивать частоту запросов и обрабатывать исключения.
Разобранный пример: Источник: jsonplaceholder.typicode.com. Сценарий: загрузка 5 постов, объединение с авторами, фильтрация пустых полей, экспорт в CSV.
Программный код:

```
import requests, pandas as pd
BASE_URL = "https://jsonplaceholder.typicode.com"
posts = requests.get(f"{BASE_URL}/posts?_limit=5").json()
users = {u['id']: u['name'] for u in requests.get(f"{BASE_URL}/users"
    ).json()}
data = [{"id": p['id'], "author": users.get(p['userId'], "Unknown"),
    "title": p['title'], "body_length": len(p['body']), "fetch_time":
    pd.Timestamp.now()} for p in posts]
df = pd.DataFrame(data)
df.to_csv("posts_export.csv", index=False, encoding="utf-8")
print("                                                    .")
```

Индивидуальное задание: Напишите скрипт загрузки данных о курсах валют или погоды. Реализуйте повторные попытки при ошибке 503, логирование и сохранение в JSON Lines.
Вопросы для самопроверки: 1) Разница парсинга HTML и REST API? 2) Как реализовать exponential backoff? 3) Почему важно проверять лицензии данных?

4 Предварительная обработка и очистка данных (Data Preprocessing)

Цель: Освоить методы выявления и устранения аномалий, обработки пропусков, кодирования категориальных признаков и масштабирования.
Задачи:

- Проанализировать распределение признаков и выявить пропуски;
- Применить стратегии импутации и обработки выбросов (IQR, Z-score);
- Выполнить кодирование (One-Hot, Label) и нормализацию;
- Сформировать готовый датасет.

Теоретическое описание: Реальные данные содержат шум и пропуски (MCAR, MAR, MNAR). Выбросы выявляются через IQR. Кодирование и масштабирование необходимы для совместимости с математическими моделями.
Разобранный пример: Датасет с пропусками в возрасте и доходе. Медианная импутация, удаление строк с `income < 0`, One-Hot для региона, StandardScaler для чисел.
Программный код:

```
import pandas as pd, numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
df = pd.DataFrame({'age': [25, np.nan, 45, 30, 120], 'income':
    [50000, 75000, -1000, 60000, 80000], 'region': ['North', 'South',
    'North', 'West', np.nan]})
df = df[(df['age'] > 16) & (df['age'] < 100) & (df['income'] > 0)]
preprocessor = ColumnTransformer(transformers=[('num', Pipeline([('
    impute', SimpleImputer(strategy='median')), ('scale',
    StandardScaler())]), ['age', 'income']), ('cat', Pipeline([('
    impute', SimpleImputer(strategy='most_frequent')), ('encode',
    OneHotEncoder(sparse_output=False))]), ['region'])])
X_clean = preprocessor.fit_transform(df)
print("                                :\n", X_clean)
```

Индивидуальное задание: Загрузите датасет Titanic. Сравните медианную и KNN-импутацию. Оцените влияние на дисперсию признаков.
Вопросы для самопроверки: 1) Когда удалять строки, а когда импутировать? 2) Почему One-Hot может вызвать «Curse of Dimensionality»? 3) В каких алгоритмах масштабирование обязательно?

5 Статистический анализ и разведочный анализ данных (EDA)

Цель: Применить методы описательной и инференциальной статистики для выявления закономерностей и проверки гипотез.
Задачи:

- Рассчитать описательные статистики и построить распределения;
- Оценить корреляции и выявить мультиколлинеарность;
- Провести проверку гипотез (t-тест, ANOVA);
- Сформулировать выводы.

Теоретическое описание: EDA — итеративный процесс изучения данных. Корреляция Пирсона измеряет линейную зависимость. p-value < 0.05 указывает на отказ от нулевой гипотезы.
Разобранный пример: Датасет: цены на жильё. Анализ: гистограммы, матрица корреляций. Гипотеза: средняя цена в районе A выше, чем в B. t-тест дал $p = 0.012 \rightarrow$ различие значимо.
Программный код:

```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt, seaborn as sns
from scipy import stats
np.random.seed(42)
df = pd.DataFrame({'area': np.random.normal(60, 15, 200), 'price': np
    .random.normal(5e6, 1.2e6, 200), 'district': np.random.choice(['A'
    , 'B'], 200)})
df['price'] += df['area'] * 50000
corr = df[['area', 'price']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
t_stat, p_val = stats.ttest_ind(df[df['district']=='A']['price'], df[
    df['district']=='B']['price'])
print(f"t={t_stat:.2f}, p={p_val:.4f}")
```

Индивидуальное задание: Проведите EDA маркетингового датасета. Проверьте гипотезу о влиянии канала на конверсию с помощью ANOVA.
Вопросы для самопроверки: 1) Почему медиана устойчивее среднего? 2) Как интерпретировать p-value? 3) Когда использовать непараметрические тесты?

6 Визуализация данных и построение интерактивных дашбордов

Цель: Освоить принципы визуальной коммуникации и создать интерактивный дашборд.
Задачи:

- Изучить правила выбора типов диаграмм;
- Реализовать интерактивные графики с Plotly;
- Спроектировать макет дашборда с фильтрами;
- Развернуть веб-приложение.

Теоретическое описание: Визуализация преобразует числа в образы. Принцип «data-ink ratio» требует максимизации полезной информации. Интерактивные дашборды позволяют исследовать сценарии в реальном времени.
Разобранный пример: Данные: ежеквартальные продажи. Дашборд: Streamlit с фильтрами по году/категории, автоматическое обновление KPI.
Программный код (dashboard.py):

```
import streamlit as st
import pandas as pd, plotly.express as px
st.set_page_config(page_title="Sales Dashboard", layout="wide")
st.title("
                ")
df = pd.read_csv("sales_data.csv")
col1, col2 = st.columns(2)
year = col1.selectbox("      ", sorted(df['year'].unique()))
category = col2.multiselect("                  ", df['category'].
   unique(), default=df['category'].unique())
filtered = df[(df['year']==year) & (df['category'].isin(category))]
fig_line = px.line(filtered.groupby('month')['revenue'].sum().
   reset_index(), x='month', y='revenue', title="
                ")
fig_bar = px.bar(filtered.groupby('category')['revenue'].sum().
   reset_index(), x='category', y='revenue', title="
                            ")
st.plotly_chart(fig_line, use_container_width=True)
st.plotly_chart(fig_bar, use_container_width=True)
```

Индивидуальное задание: Создайте дашборд успеваемости студентов с фильтрами и выявлением «зоны риска».
Вопросы для самопроверки: 1) Почему круговые диаграммы не рекомендуются для >5 категорий? 2) Как цвет влияет на восприятие? 3) Преимущество Plotly/Dash?

7 Применение базовых алгоритмов машинного обучения для прогнозирования

Цель: Реализовать цикл построения и валидации моделей МО для регрессии и классификации.
Задачи:

- Разделить выборку;
- Обучить модели (линейная регрессия, случайный лес);
- Оценить качество метриками (RMSE, MAE, F1, ROC-AUC);
- Проанализировать важность признаков.

Теоретическое описание: МО находит закономерности без явного программирования. Ключевой этап — разделение данных. $RMSE = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}$. Интерпретируемость обеспечивается feature importance.
Разобранный пример: Прогноз стоимости авто. Модель: RandomForestRegressor. Метрики: $R^2 = 0.89$, RMSE = 1.2 тыс. \$. Важные признаки: пробег, год.
Программный код:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
df = pd.read_csv("cars.csv")
X, y = df[['year', 'mileage', 'engine_volume']], df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(f"R2: {r2_score(y_test, y_pred):.3f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(pd.Series(model.feature_importances_, index=X.columns).
    sort_values(ascending=False))
```

Индивидуальное задание: Решите задачу классификации (отток клиентов). Сравните логистическую регрессию и градиентный бустинг, постройте ROC-кривую.
Вопросы для самопроверки: 1) Что такое переобучение? 2) Почему Accuracy не подходит для несбалансированных данных? 3) Как кросс-валидация улучшает оценку?

8 Обеспечение безопасности и аудита в процессах обработки информации

Цель: Изучить методы защиты данных, контроля доступа и ведения аудиторских журналов.
Задачи:

- Освоить хеширование, шифрование и псевдонимизацию;
- Реализовать ролевую модель доступа (RBAC);
- Настроить логирование операций;
- Провести анализ соответствия 152-ФЗ/GDPR.

Теоретическое описание: Безопасность строится на принципе CIA. Хеширование (SHA-256) обеспечивает целостность. Шифрование защищает конфиденциальность. Аудит фиксирует события доступа. Регуляторы требуют минимизации данных и права на удаление.
Разобранный пример: Медицинские записи. Защита: хеширование email, маскация ФИО. Аудит: запись всех операций в лог с timestamp. Результат: данные пригодны для анализа, идентификаторы скрыты.
Программный код:

```
import pandas as pd, hashlib, logging
logging.basicConfig(filename='audit.log', level=logging.INFO, format=
    '%(asctime)s | %(levelname)s | %(message)s')
def mask_name(name): return name[0] + '*' * (len(name)-1) if len(name
    )>1 else '*'
def hash_email(email): return hashlib.sha256(email.encode()).
    hexdigest()
df = pd.DataFrame({'full_name': ['                    .   .', '
                    .   .'], 'email': ['ivan@test.ru', 'petrova@test.
    ru'], 'diagnosis': ['Asthma', 'Diabetes']})
df['full_name'] = df['full_name'].apply(mask_name)
df['email_hash'] = df['email'].apply(hash_email)
df = df.drop(columns=['email'])
df.to_csv('safe_data.csv', index=False)
logging.info(f"Processed {len(df)} records.")
print("
                      .")
```

Индивидуальное задание: Разработайте аудит для HR-базы. Реализуйте шифрование поля «зарплата» через `cryptography` и журнал изменений.
Вопросы для самопроверки: 1) Разница хеширования и шифрования? 2) Что такое псевдонимизация? 3) Требования стандартов к лог-файлам?

Список литературы

[1] McKinney W. Python for Data Analysis: Data Wrangling with pandas. 3rd ed. O'Reilly, 2022.

[2] Grus J. Data Science from Scratch. 2nd ed. O'Reilly, 2019.

[3] James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. Springer, 2021.

[4] Few S. Show Me the Numbers: Designing Tables and Graphs to Enlighten. Analytics Press, 2012.

[5] Stansifer R. Database Management Systems: Concepts and Design. Pearson, 2020.

[6] Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union, 2016.

[7] Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».

[8] pandas development team. pandas documentation. URL: `https://pandas.pydata.org/docs/`.

[9] Pedregosa F. et al. Scikit-learn: Machine Learning in Python. JMLR, 2011.

[10] Python Software Foundation. Python 3.11 Documentation. URL: `https://docs.python.org/3/`.