**Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение**

**«Академия управления городской средой, градостроительства и печати»**

**УТВЕРЖДАЮ**

**Заместитель директора**

**по учебно-методической работе**

**_____О.В. Фомичёва**

**«____» _____ 20_____г.**

# Методические рекомендации по организации и проведению практических занятий

## *ПМ.01 «РАЗРАБОТКА КОДА ДЛЯ ОБУЧЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»*
## *МДК 01.01. Разработка программных модулей в системах искусственного интеллекта*

**для специальности**

**специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта**

**Форма обучение –очная**

**Санкт-Петербург**
**2025**

Разработчик: Ипатова С.В./Оболенская Е.Г., методисты СПб ГБПОУ АУГСГиП

Одобрены на заседании цикловой комиссии

Общетехнических дисциплин и компьютерных технологий

Протокол № 4

09.12.2025 г.

Председатель цикловой комиссии:

Шурухина И.Е.

В результате изучения профессионального модуля обучающихся должен освоить основной вид деятельности ВД1. «Разработка кода для обучения искусственного интеллекта» и соответствующие ему общие компетенции и профессиональные компетенции:

1.1.1. Перечень общих компетенций

| *Код* | Наименование общих компетенций |
|---|---|
| **ОК 01** | Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам |
| **ОК 02** | Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности |
| **ОК 03** | Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях |
| **ОК 04** | Эффективно взаимодействовать и работать в коллективе и команде |
| **ОК 05** | Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста |
| **ОК 06** | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения |
| **ОК 07** | Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях |
| **ОК 08** | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности |
| **ОК 09** | Пользоваться профессиональной документацией на государственном и иностранном языках |

1.1.2. Перечень профессиональных компетенций

| *Код* | Наименование видов деятельности и профессиональных компетенций |
|---|---|
| **ВД 1** | Разработка кода для искусственного интеллекта |
| **ПК 1.1** | Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием. |
| **ПК 1.2** | Разрабатывать программные модули в соответствии с техническим заданием. |
| **ПК 1.3** | Оформлять программный код в соответствии с техническим заданием. |
| **ПК 1.4** | Использовать систему контроля версий программного кода с учетом обеспечения возможности организации групповой разработки. |
| **ПК 1.5** | Выполнять отладку программных модулей с использованием специализированных программных средств. |
| **ПК 1.6** | Выполнять тестирование программного кода. |
| **ПК1.7.** | Составлять тестовые сценарии. |

1.1.3. В результате освоения профессионального модуля обучающийся должен:

| | |
|---|---|
| **Иметь практический опыт** | – Разработки, оптимизации и оценки сложности алгоритмов для ИИ-программ.<br>– Использования библиотек и инструментов для работы с алгоритмами и данными (например: Pandas, NumPy, Scikit-learn).<br>– Применения структур данных (деревья, графы, списки) для реализации алгоритмов. |

| | |
|---|---|
| | – Разработки модульных ИИ-систем, соответствующих требованиям производительности и безопасности.<br>– Внедрения разработанных ИИ-модулей в комплексные программные системы.<br>– Оптимизации кода и работы с интерфейсами для взаимодействия между модулями.<br>– Оформления, документирования и структурирования кода для последующей поддержки.<br>– Использования инструментов статического анализа кода для выявления ошибок и улучшения качества.<br>– Работы с системами документирования кода (например, Doxygen, Sphinx).<br>– Управления проектами с использованием систем контроля версий для организации командной работы.<br>– Разрешения конфликтов при слиянии веток и использования pull request для рецензирования кода.<br>– Настройки процессов CI/CD для автоматического тестирования и развертывания кода.<br>– Отладки программных модулей с использованием пошаговой проверки.<br>– Применения методов логирования и профилирования производительности.<br>– Использования специальных средств для отладки многопоточных программ.<br>– Выполнения статического тестирования программного кода на предмет выявления ошибок/дефектов алгоритмов, в том числе – на наличие обработки исключений<br>– Выполнения тестирования программных модулей в соответствии в тест-планом<br>– Генерирования тестовых данных<br>– Выполнения интеграционного тестирования в соответствии с заданием<br>– Выполнения регрессионного тестирования в соответствии с заданием.<br>– Работы с CI/CD пайплайнами для автоматизации тестирования.<br>– Разработки тестовых сценариев в соответствии с тестовым планом (тестирование производительности, надежности, UI-тестирование), в том числе с применением средств автоматизации проектирования.<br>– Разработки тестовых пакетов и заданий на выполнение тестирования.<br>– Оценки тестовых данных на предмет покрытия строк и покрытия ветвей, выполнения валидации данных.<br>– Автоматизации создания и выполнения тестовых сценариев. |
| **Уметь** | – Анализировать технические задания и выявлять требования к алгоритмам.<br>– Применять методы алгоритмизации для решения задач программирования.<br>– Разрабатывать оптимальные алгоритмы для решения задач в области ИИ. |

| | |
|---|---|
| | – Реализовывать программные модули на основе требований технического задания.<br>– Соблюдать при разработке принципы «чистого кода».<br>– Использовать стандартные библиотеки и фреймворки для ускорения разработки.<br>– Оформлять код в соответствии с принятыми стандартами и требованиями.<br>– Документировать разработанный программный код.<br>– Соблюдать соглашения о наименованиях переменных, функций и классов (например, PEP8 для Python).<br>– Работать с системами контроля версий для управления проектами.<br>– Организовывать совместную работу над проектом через ветки разработки и слияние изменений.<br>– Разрешать конфликты при слиянии кода.<br>– Использовать инструменты для отладки программного кода.<br>– Идентифицировать и исправлять ошибки в программе.<br>– Применять методы логирования для анализа выполнения программ.<br>– Проводить различные виды тестирования (юнит-тестирование, интеграционное тестирование).<br>– Выполнять настройки окружения т подготовку тестовых данных<br>– Фиксировать результаты выполнения тестов и подготавливать отчеты о результатах тестов.<br>– Определять уровень критичности дефектов.<br>– Разрабатывать автоматизированные тесты для тестирования модулей и/или отдельных функций<br>– Восстанавливать окружение и тесты после сбоя<br>– Проектировать тестовые сценарии на основе тестовых планов.<br>– Разрабатывать тестовые пакеты и задания на выполнение тестирования.<br>– Использовать шаблоны для написания тест-кейсов.<br>– Оценивать риски при отборе тестов для регрессионного тестирования.<br>– Оценивать тесты на соответствие целям тестирования. |
| **Знать** | – Основные методы и подходы к построению алгоритмов (типовые поисковые алгоритмы, жадные алгоритмы, динамическое программирование, рекурсивные подходы).<br>– Принципы эффективной обработки данных.<br>– Языки программирования, применяемые для разработки алгоритмов.<br>– Принципы модульного программирования.<br>– Языки программирования для разработки модулей.<br>– Стандартные фреймворки и библиотеки для работы с ИИ.<br>– Основные принципы чистого кода (Clean Code).<br>– Стандарты и практики документирования программного обеспечения.<br>– Инструменты для автоматической проверки качества кода (например, PyLint, ESLint).<br>– Принципы работы распределенных систем контроля версий.<br>– Основные команды и операции в системах контроля версий |

| | |
|---|---|
| | (например: commit, pull, push, merge).<br>– Методы разрешения конфликтов в ходе групповой разработки.<br>– Принципы работы отладчиков и логирования.<br>– Способы выявления ошибок в программе (отладка по шагам, точки останова).<br>– Инструменты для отладки кода (например, PyCharm, Visual Studio Debugger).<br>– Техники выполнения тестовых прогонов.<br>– Инструменты и среды выполнения тестирования<br>– Языки разработки автоматизированных тестов<br>– Инструменты для тестирования программного кода.<br>– Правила выполнения отчетов о тестировании<br>– Цели, задачи и виды тестирования. Понятие стратегии тестирования.<br>– Жизненный цикл дефекта.<br>– Основы тест-дизайна: тестовый сценарий, тестовый пакет, чек-лист, основные шаблоны.<br>– Основные инструменты проектирования тестов.<br>– Методы и подходы к написанию тестов (Test-Driven Development, Behavior-Driven Development). |

## *Практические работы*

| тема | название ПР | часы |
|---|---|---|
| МДК 01.01 Разработка программных модулей в системах искусственного интеллекта | | |
| **Тема 1.1. Введение в искусственный интеллект и его направления** | **Практическая работа №1.** Анализ примеров успешных решений на основе ИИ. | 8 |
| | **Практическая работа №2.** Создание базовой модели ИИ для классификации данных. | 8 |
| **Тема 1.2. Методы сбора и предобработки данных** | **Практическая работа №3**. Сбор данных с использованием веб-скрапинга и API. | 6 |
| | **Практическая работа №4.** Предобработка данных для машинного обучения: очистка, нормализация, кодирование. | 8 |
| **Тема 1.3. Основы алгоритмов машинного обучения** | **Практическая работа №5.** Реализация линейной регрессии на реальных данных. | 6 |
| | **Практическая работа №6.** Применение кластеризации для сегментации данных. | 6 |
| **Тема 1.4. Оценка качества моделей и улучшение алгоритмов** | **Практическая работа №7.** Оценка качества модели с использованием ROC-кривой и F-меры. | 6 |
| | **Практическая работа №8.** Настройка гиперпараметров модели с использованием GridSearchCV. | 6 |
| **Тема 1.5. Глубокое обучение и нейронные сети** | **Практическая работа №9.** Реализация многослойного перцептрона (MLP) для задачи классификации. | 6 |
| | **Практическая работа №10.** Создание сверточной нейронной сети для распознавания изображений. | 6 |
| | **Практическая работа №11.** Реализация рекуррентной нейронной сети для анализа временных рядов. | 6 |
| **Тема 1.6. Проектирование ИИ-систем** | **Практическая работа №12**. Проектирование архитектуры ИИ-системы с учетом модульности и масштабируемости. | 8 |
| | **Практическая работа №13**. Контейнеризация ИИ-модели с использованием Docker. | 8 |
| | **Практическая работа №14.** Развертывание ИИ-системы в Kubernetes. | 8 |
| | | 96 |

# МДК 01.01 Разработка программных модулей в системах искусственного интеллекта

## Практическая работа №1. Анализ примеров успешных решений на основе ИИ.

***Цель работы:*** изучить реальные кейсы применения искусственного интеллекта, выявить ключевые преимущества и закономерности успешного внедрения ИИ-технологий.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретические основы**

**Искусственный интеллект (ИИ)** — комплекс алгоритмов и систем, способных:

- анализировать большие массивы данных;
- выявлять закономерности;
- принимать решения, сопоставимые с человеческими;
- обучаться и адаптироваться к новым условиям.

Ключевые направления применения ИИ:

- прогнозирование и аналитика;
- автоматизация рутинных процессов;
- обработка естественного языка;
- распознавание образов и паттернов;
- оптимизация ресурсов и рисков.

**Примеры успешных внедрений ИИ**

1. **Tesla (автомобильная промышленность)**
   - **Задача:** оптимизация производственных процессов и цепочек поставок.
   - **Решение:** алгоритмы машинного обучения анализируют данные с производственной линии.
   - **Результаты:**

- сокращение времени выпуска новых моделей;
- прогнозирование потребности в комплектующих;
- адаптация процессов к изменяющимся условиям.

2. **IBM Watson (IT-проекты)**
   - **Задача:** управление разработкой ПО и минимизация рисков.
   - **Решение:** система ИИ анализирует исторические данные проектов.
   - **Результаты:**
     - точное прогнозирование потенциальных проблем;
     - оптимизация распределения ресурсов;
     - повышение эффективности команд.
3. **Умные города (транспортная инфраструктура)**
   - **Задача:** снижение пробок и повышение безопасности дорожного движения.
   - **Решение:** ИИ обрабатывает данные со светофоров, камер и метеостанций.
   - **Результаты:**
     - динамическое управление светофорами;
     - прогноз трафика в реальном времени;
     - оперативное реагирование на аварии.
4. **Сельское хозяйство (агротехнологии)**
   - **Задача:** повышение урожайности и снижение затрат.
   - **Решение:** роботы с ИИ контролируют состояние растений и почвы.
   - **Результаты:**
     - точечное внесение удобрений и пестицидов;
     - раннее выявление болезней растений;
     - автоматизация прополки.
5. **Здравоохранение (диагностика)**
   - **Задача:** ускорение и повышение точности постановки диагнозов.
   - **Решение:** системы типа IBM Watson анализируют миллионы медицинских записей.
   - **Результаты:**
     - выявление заболеваний на ранних стадиях;
     - персонализированные рекомендации по лечению;
     - снижение нагрузки на врачей.
6. **Ритейл и сервис (клиентский опыт)**
   - **Задача:** персонализация предложений и поддержка клиентов.
   - **Решение:** чат-боты и рекомендательные системы на базе ИИ.
   - **Результаты:**
     - 24/7 обслуживание клиентов;
     - увеличение конверсии за счёт таргетированных предложений;
     - анализ предпочтений покупателей.

**Общие закономерности успеха**

1. **Качество данных**
   - успешные проекты требуют больших объёмов актуальных и структурированных данных;
   - важна предварительная очистка и валидация информации.
2. **Интеграция с бизнес-процессами**

- ИИ-решения должны встраиваться в существующие рабочие потоки;
- необходима адаптация процессов под возможности ИИ.

3. **Человеко-ориентированный подход**
   - ИИ дополняет, а не заменяет специалистов;
   - финальные решения часто остаются за человеком.
4. **Итеративное развитие**
   - системы обучаются и совершенствуются с каждым новым проектом;
   - важна обратная связь от пользователей.
5. **Экономическая эффективность**
   - сокращение затрат на рутинные операции;
   - повышение скорости принятия решений;
   - минимизация ошибок и рисков.

**Выводы**

Анализ кейсов показывает, что успешное применение ИИ базируется на:

- чётком определении бизнес-задачи;
- доступе к качественным данным;
- кросс-функциональной команде (аналитики, инженеры, предметные эксперты);
- готовности компании к цифровой трансформации.

**Перспективы развития:**

- усиление роли генеративных моделей (например, для создания контента);
- интеграция ИИ с IoT и блокчейн-технологиями;
- развитие объяснимого ИИ (Explainable AI) для повышения доверия пользователей.

**Контрольные вопросы**

1. Какие ключевые факторы определяют успех внедрения ИИ в бизнес-процессы?
2. Приведите 2–3 примера отраслей, где ИИ пока не получил широкого распространения. Почему?
3. В чём заключается роль человека в системах с ИИ-поддержкой?
4. Какие риски могут возникнуть при некорректном использовании ИИ?

**Задание для самостоятельной работы**

Выберите одну из отраслей (образование, юриспруденция, логистика), найдите 1–2 примера успешного внедрения ИИ и подготовьте краткий анализ по схеме:

- задача, которую решал ИИ;
- технология/алгоритм;
- достигнутые результаты;
- возможные ограничения.

***Оформление результатов работы***

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

***Содержание отчёта***

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и

кодированию.

**Ответить на контрольные вопросы. (устно)**
***Сформулировать выводы по результатам работы.***
***Сдать и защитить работу.***

## Практическая работа №2. Создание базовой модели ИИ для классификации данных.

***Цель работы:*** освоить базовый цикл разработки модели машинного обучения для задачи классификации: от подготовки данных до оценки качества модели.
данных.
***Задачи:***
- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***
1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***
1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Необходимое ПО и библиотеки**
Для выполнения работы потребуется:
- Python 3.8+
- Jupyter Notebook / Google Colab
- Библиотеки:
- pip install numpy pandas scikit-learn matplotlib seaborn

**Шаг 1. Подготовка среды и данных**
1. Создайте проектную структуру:

```
project_ml/
├── data/            # исходные данные
├── notebooks/       # Jupyter-ноутбуки
├── src/             # код
```

```
6. │   ├── data_prep.py   # предобработка
7. │   ├── model.py       # модель
8. │   └── visualize.py   # визуализация
9. ├── requirements.txt   # зависимости
10.          └── README.md         # описание
```

11. Загрузите набор данных. Для практики подойдёт:

- Iris dataset (классификация видов ирисов)
- Titanic dataset (предсказание выживаемости)
- Breast Cancer dataset (диагностика рака)

```
python
from sklearn.datasets import load_iris
data = load_iris()
X, y = data.data, data.target
```

**Шаг 2. Анализ и предобработка данных**

1. Изучите данные:

```
python
import pandas as pd
import numpy as np

df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print(df.head())
print(df.describe())
```

2. Проверьте на пропуски и выбросы:

```
python
print(df.isnull().sum())
# Визуализация выбросов
import seaborn as sns
sns.boxplot(data=df.iloc[:, :-1])
```

3. Разделите на признаки (X) и целевую переменную (y):

```
python
X = df.drop('target', axis=1)
y = df['target']
```

4. Нормализуйте признаки:

```
python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

**Шаг 3. Разделение данных**

Разделите данные на обучающую и тестовую выборки (70%/3 gef=“30%”):

```
python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y
)
```

Шаг 4. Выбор и обучение модели

1. Выберите алгоритм классификации. Для старта рекомендуются:

- Логистическая регрессия
- Метод опорных векторов (SVM)
- Случайный лес (Random Forest)

2. Пример обучения модели:

```
python
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

# Инициализация модели
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Обучение
model.fit(X_train, y_train)
```

**Шаг 5. Оценка качества модели**

1. Сделайте предсказания на тестовой выборке:

```
python
y_pred = model.predict(X_test)
```

2. Рассчитайте метрики качества:

```
python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
print("F1-score:", f1_score(y_test, y_pred, average='macro'))
```

3. Визуализируйте матрицу ошибок:

```
python
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
```

**Шаг 6. Интерпретация результатов**

1. Проанализируйте отчёт классификации:

```
python
print(classification_report(y_test, y_pred))
```

2. Определите классы с наихудшей классификацией.
3. Оцените важность признаков (для Random Forest):

```
python
feature_importance = pd.DataFrame({
    'feature': data.feature_names,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
print(feature_importance)
```

**Шаг 7. Сохранение модели**

Сохраните обученную модель для дальнейшего использования:

```
python
import joblib

joblib.dump(model, 'model_rf.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

**Контрольные вопросы**

1. Какие этапы включает цикл разработки модели классификации?
2. Зачем нужно разделять данные на обучающую и тестовую выборки?
3. В чём разница между accuracy, precision и recall?
4. Почему важно проводить нормализацию признаков?
5. Как интерпретировать матрицу ошибок?

**Варианты заданий (на выбор)**

1. Классифицировать виды ирисов (Iris dataset) с точностью > 95 %.
2. Предсказать выживаемость на Титанике (Titanic dataset) с F1-score > 0.7.
3. Диагностировать рак молочной железы (Breast Cancer dataset) с recall > 0.85 для положительного класса.

**Требования к отчёту**

Отчёт должен содержать:

- Описание выбранного датасета
- Код с комментариями по каждому этапу
- Таблицы и графики результатов
- Интерпретацию метрик качества
- Выводы о работе модели и возможных улучшениях

***Оформление результатов работы***

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

***Содержание отчёта***

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

**Ответить на контрольные вопросы. (устно)**

***Сформулировать выводы по результатам работы.***

***Сдать и защитить работу.***

## Практическая работа №3. Сбор данных с использованием веб-скрапинга и API.

***Цель работы:*** освоить методы автоматизированного сбора данных из веб-источников:

- через веб-скрапинг (парсинг HTML-страниц);
- через API веб-сервисов.

сбор информации с веб-сайтов и интегрировать функционал других сервисов в собственные

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.

2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.

4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).

5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая часть**

**1. Веб-скрапинг**

**Веб-скрапинг** — автоматизированный сбор данных с веб-сайтов путём анализа HTML-кода страниц.

**Основные инструменты (Python):**

- `requests` — отправка HTTP-запросов и получение ответов;
- `BeautifulSoup` — парсинг и навигация по HTML-структуре;
- `Scrapy` — фреймворк для масштабного скрапинга с поддержкой асинхронности;
- `Selenium` / `Playwright` — автоматизация браузера для работы с динамическим контентом (JavaScript).

**Этапы скрапинга:**

1. Отправка HTTP-запроса к URL.
2. Получение HTML-ответа.
3. Анализ структуры страницы (селекторы: CSS, XPath).
4. Извлечение нужных данных.
5. Сохранение результатов (CSV, JSON, база данных).

**Важные ограничения:**

- соблюдение `robots.txt` и условий использования сайта;
- ограничение частоты запросов (rate limiting);
- обработка ошибок и таймаутов;
- использование заголовков и куков при необходимости.

**2. Работа с API**

**API (Application Programming Interface)** — интерфейс для программного взаимодействия с сервисом.

**Типы API:**

- REST API (наиболее распространённый);
- GraphQL;
- SOAP и др.

**Основные методы HTTP:**

- `GET` — получение данных;
- `POST` — отправка данных;
- `PUT` / `PATCH` — обновление;
- `DELETE` — удаление.

**Формат данных:**

- JSON (чаще всего);
- XML;

- Protobuf и др.

**Шаги работы с API:**

1. Регистрация и получение API-ключа (если требуется).
2. Изучение документации API (endpoints, параметры, лимиты).
3. Формирование запроса с нужными параметрами.
4. Обработка ответа (проверка статуса, парсинг данных).
5. Сохранение результатов.

**Практическая часть**

**Задание 1. Веб-скрапинг страницы**

**Цель:** извлечь данные с публичной веб-страницы.

**Шаг 1.** Выберите целевой сайт (например, новостной портал, каталог товаров).

**Шаг 2.** Установите зависимости:

```
bash
pip install requests beautifulsoup4 lxml
```

**Шаг 3.** Напишите скрипт на Python:

```
python
import requests
from bs4 import BeautifulSoup
import csv

# 1. Отправляем запрос
url = "https://example.com/news"
response = requests.get(url)
response.raise_for_status()  # Проверяем статус

# 2. Парсим HTML
soup = BeautifulSoup(response.content, 'html.parser')

# 3. Находим элементы (пример для новостей)
articles = []
for item in soup.select('.news-item'):
    title = item.select_one('.title').get_text(strip=True)
    link = item.select_one('a')['href']
    date = item.select_one('.date').get_text(strip=True)
    articles.append({'title': title, 'link': link, 'date': date})

# 4. Сохраняем в CSV
with open('news.csv', 'w', newline='', encoding='utf-8') as f:
    writer = csv.DictWriter(f, fieldnames=['title', 'link', 'date'])
    writer.writeheader()
    writer.writerows(articles)
```

**Шаг 4.** Проверьте результат: файл `news.csv` должен содержать извлечённые данные.

**Задание 2. Работа с публичным API**

**Цель:** получить данные через API и сохранить их.

**Шаг 1.** Выберите публичный API (например, , , ).

**Шаг 2.** Получите API-ключ (если требуется) и изучите документацию.

**Шаг 3.** Установите зависимости:

```
bash
pip install requests
```

**Шаг 4.** Напишите скрипт:

```
python
import requests
```

```
import json

# 1. Настройки API
api_url = "https://jsonplaceholder.typicode.com/posts"
headers = {
    "Content-Type": "application/json"
}

# 2. Отправляем GET-запрос
response = requests.get(api_url, headers=headers)
response.raise_for_status()

# 3. Парсим JSON
data = response.json()

# 4. Фильтруем нужные поля (пример)
posts = []
for post in data[:10]:  # Берём первые 10 записей
    posts.append({
        'id': post['id'],
        'title': post['title'],
        'body': post['body'][:100]  # Первые 100 символов
    })

# 5. Сохраняем в JSON
with open('posts.json', 'w', encoding='utf-8') as f:
    json.dump(posts, f, ensure_ascii=False, indent=2)
```

**Шаг 5.** Проверьте результат: файл `posts.json` должен содержать данные из API.

**Требования к отчёту**

Подготовьте отчёт, включающий:

1. **Цель работы** и краткое описание задач.
2. **Список использованных инструментов** (библиотеки, API).
3. **Код скриптов** с комментариями.
4. **Примеры выходных данных** (фрагменты CSV/JSON).
5. **Анализ результатов**:
   - какие данные удалось собрать;
   - с какими трудностями столкнулись;
   - как их преодолевали.
6. **Выводы** по проделанной работе.

**Критерии оценки**

- Корректность работы скриптов.
- Качество извлечённых данных (точность, полнота).
- Соблюдение этических норм (robots.txt, rate limiting).
- Оформление отчёта (чёткость, структура, комментарии в коде).

**Дополнительные задания (по желанию)**

1. Реализуйте обработку пагинации (переход по страницам сайта/API).
2. Добавьте логирование ошибок и прогресс-бар.
3. Сохраните данные в базу данных (SQLite, PostgreSQL).
4. Сравните скорость скрапинга через HTML и API для одного источника.

***Оформление результатов работы***

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

***Содержание отчёта***

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

**Ответить на контрольные вопросы. ( устно)**

***Сформулировать выводы по результатам работы.***

***Сдать и защитить работу.***

## Практическая работа №4. Предобработка данных для машинного обучения: очистка, нормализация, кодирование.

***Цель работы:*** освоить основные методы предобработки данных для машинного обучения: очистку от пропусков и выбросов, нормализацию (масштабирование) и кодирование категориальных признаков.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая часть**

**Предобработка данных** — ключевой этап построения ML-модели, от которого зависит её качество и устойчивость. Включает:

- очистку (удаление/заполнение пропусков, выбросов, дубликатов);

- трансформацию (приведение типов, создание новых признаков);
- нормализацию (масштабирование признаков);
- кодирование (преобразование категориальных данных в числовые).

**1. Очистка данных**

**Пропущенные значения** (`NaN`, `None`, пустые строки) обрабатывают так:

- удаление строк/столбцов (если пропусков мало);
- заполнение средним/медианой (для числовых);
- заполнение наиболее частым значением (для категориальных);
- интерполяция (для временных рядов);
- алгоритмы вроде `KNNImputer` (заполнение на основе ближайших соседей).

**Выбросы** выявляют через:

- Z-score ($|z|>3$);
- межквартильный размах (IQR): значения вне $[Q1-1{,}5\cdot IQR;\ Q3+1{,}5\cdot IQR]$;
- визуальные методы (boxplot, scatterplot).

**Дубликаты** удаляют по всем столбцам или ключевым полям.

**2. Нормализация (масштабирование)**

Приводит признаки к единому масштабу. Основные методы:

- **Min-Max Scaling** (от 0 до 1):

$Xnorm=Xmax-XminX-Xmin$

- **Standardization** (нулевое среднее, единичная дисперсия):

$Xstd=\sigma X-\mu$

- **Robust Scaling** (на основе медианы и IQR) — устойчив к выбросам.

**3. Кодирование категориальных данных**

- **One-Hot Encoding** — создаёт бинарные столбцы для каждой категории (подходит для неиерархических признаков).
- **Label Encoding** — присваивает категориям числа (0, 1, 2, …); опасен для алгоритмов, воспринимающих порядок.
- **Target Encoding** — заменяет категорию на среднее значение целевой переменной (требует осторожности из-за переобучения).

**Практическая часть**

**Шаг 1. Загрузка и анализ данных**

```
python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder,
LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

# Загрузка данных
df = pd.read_csv('data.csv')

# Базовый анализ
print(df.info())
print(df.describe())
print(df.isna().sum())  # Пропуски
```

**Шаг 2. Очистка**

**а) Удаление дубликатов:**

```
python
df.drop_duplicates(inplace=True)
```

**б) Заполнение пропусков:**

python

```
# Для числовых: медиана
imputer_median = SimpleImputer(strategy='median')
df[['age', 'salary']] = imputer_median.fit_transform(df[['age', 'salary']])

# Для категориальных: наиболее частое значение
imputer_most_frequent = SimpleImputer(strategy='most_frequent')
df[['city']] = imputer_most_frequent.fit_transform(df[['city']])
```

**в) Удаление выбросов (по Z-score):**

python

```
from scipy import stats
z_scores = np.abs(stats.zscore(df[['salary']]))
df = df[(z_scores < 3).all(axis=1)]
```

**Шаг 3. Нормализация**

python

```
# Min-Max Scaling
scaler_minmax = MinMaxScaler()
df[['salary_norm']] = scaler_minmax.fit_transform(df[['salary']])

# Standardization
scaler_std = StandardScaler()
df[['age_std']] = scaler_std.fit_transform(df[['age']])
```

**Шаг 4. Кодирование**

python

```
# One-Hot Encoding
encoder_ohe = OneHotEncoder(sparse_output=False)
ohe_features = encoder_ohe.fit_transform(df[['city']])
ohe_df = pd.DataFrame(ohe_features,
columns=encoder_ohe.get_feature_names_out(['city']))
df = pd.concat([df, ohe_df], axis=1)

# Label Encoding
encoder_label = LabelEncoder()
df['department_encoded'] = encoder_label.fit_transform(df['department'])
```

**Шаг 5. Проверка результатов**

python

```
print(df.isna().sum())  # Должно быть 0
print(df.describe())    # Проверить масштабы признаков
print(df.dtypes)        # Убедиться в корректности типов
```

**Задания для самостоятельной работы**

1. Загрузите датасет (например, `titanic.csv` из `seaborn`).
2. Проведите первичный анализ: выведите количество пропусков, типы данных, статистические характеристики.
3. Удалите дубликаты (если есть).
4. Заполните пропуски:
   - для числовых столбцов — медианой;
   - для категориальных — наиболее частым значением.
5. Удалите выбросы в столбце `fare` по методу IQR.
6. Нормализуйте столбцы `age` и `fare` с помощью `MinMaxScaler`.
7. Заккодируйте столбец `embarked` методом One-Hot Encoding.

8. Сохраните обработанный датасет в файл `cleaned_titanic.csv`.

**Контрольные вопросы**

1. Почему важно проводить предобработку данных перед обучением модели?
2. В чём разница между `MinMaxScaler` и `StandardScaler`? Когда какой использовать?
3. Почему `LabelEncoding` может быть опасен для категориальных признаков без естественного порядка?
4. Как выявить выбросы с помощью IQR?
5. В каких случаях стоит удалять строки с пропусками, а не заполнять их?

**Критерии оценки**

- Корректность кода (отсутствие ошибок выполнения).
- Обоснованность выбранных методов очистки/нормализации/кодирования.
- Качество итогового датасета (отсутствие пропусков, корректные типы данных).
- Ответы на контрольные вопросы (полнота, точность).

**Рекомендуемые инструменты**

- **Pandas** — загрузка, анализ, очистка данных.
- **Scikit-learn** — масштабирование, кодирование, заполнение пропусков.
- **Matplotlib/Seaborn** — визуализация пропусков и выбросов.
- **SciPy** — статистические тесты (Z-score).
- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №5. Реализация линейной регрессии на реальных данных.

***Цель работы:*** освоить методику построения и анализа линейной регрессионной модели на реальных данных, включая подготовку данных, оценку параметров, интерпретацию результатов и проверку качества модели.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий

2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.

2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.

4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).

5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Оборудование и ПО**

- MS Excel **или** Python (библиотеки: `pandas`, `numpy`, `matplotlib`, `scikit-learn`)
- Набор реальных данных (пример: цены на недвижимость, продажи товаров, метеорологические показатели и т. п.)

---

**Ход работы**

**1. Загрузка и первичная обработка данных**

1. Выберите набор данных (например, из открытых источников: Kaggle, Google Dataset Search, государственные порталы).
2. Загрузите данные в выбранное ПО:
   - В Excel: через «Данные» → «Получить данные» → «Из файла».
   - В Python:

```
python
import pandas as pd
df = pd.read_csv('ваш_файл.csv')
```

3. Проверьте структуру данных:
   - количество строк и столбцов;
   - типы данных (`int`, `float`, `object`);
   - наличие пропусков (`NaN`).

**2. Описательная статистика**

Вычислите для каждой переменной:

- среднее ($\bar{x}$);
- медиану;
- стандартное отклонение ($\sigma$);
- минимум и максимум;

- количество наблюдений.

**В Excel:** «Данные» → «Анализ данных» → «Описательная статистика».

**В Python:**

```
python
df.describe()
```

**3. Визуальный анализ данных**

Постройте:

- гистограммы распределения для каждой переменной;
- диаграмму рассеяния (scatter plot) между зависимой переменной $y$ и каждой независимой переменной $x_i$;
- матрицу корреляций (тепловую карту).

**В Python:**

```
python
import seaborn as sns
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

**4. Подготовка данных к регрессии**

1. Определите зависимую переменную ($y$) и независимые переменные ($X$).
2. Обработайте пропуски:
   - удалите строки с пропусками;
   - заполните медианой/средним.
3. Проведите масштабирование (если нужно):
   - стандартизация: $z = \sigma x - \bar{x}$;
   - нормализация: $x' = x_{max} - x_{min} x - x_{min}$.

**В Python:**

```
python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

**5. Построение модели линейной регрессии**

**Вариант А. В Excel**

1. «Данные» → «Анализ данных» → «Регрессия».
2. Укажите:
   - входной интервал $Y$;
   - входной интервал $X$;
   - уровень надёжности (95 %);
   - выходной интервал.
3. Нажмите «ОК».

**Вариант Б. В Python**

```
python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Обучение модели
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Прогноз
y_pred = model.predict(X_test)
```

**6. Анализ результатов**

1. **Коэффициенты модели**:
   - свободный член ($b_0$);
   - коэффициенты при переменных ($b_1, b_2, \ldots, b_n$).

Интерпретируйте их смысл (например, «при увеличении $x_1$ на 1 ед., $y$ растёт на $b_1$ ед.»).

2. **Статистическая значимость**:
   - $p$-значения коэффициентов (в Excel: столбец «P-значение»);
   - если $p<0,05$, коэффициент значим.
3. **Качество модели**:
   - коэффициент детерминации $R^2$ (в Excel: «R-квадрат»);
   - стандартная ошибка регрессии;
   - F-статистика (проверка значимости модели в целом).

**В Python:**

```
python
print(f"R²: {model.score(X_test, y_test)}")
print(f"Коэффициенты: {model.coef_}")
print(f"Свободный член: {model.intercept_}")
```

**7. Проверка предпосылок регрессии**

Постройте графики:

- остатки ($e_i = y_i - \hat{y}_i$) vs. предсказанные значения $\hat{y}_i$ (проверка гомоскедастичности);
- квантиль-квантиль (QQ-plot) остатков (проверка нормальности).

**В Python:**

```
python
import matplotlib.pyplot as plt
plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Предсказанные значения")
plt.ylabel("Остатки")
plt.show()
```

**8. Прогноз**

Используйте модель для предсказания $y$ на новых данных. Пример:

```
python
new_data = [[x1_new, x2_new, ..., xn_new]]
prediction = model.predict(new_data)
```

---

**Требования к отчёту**

1. **Титульный лист** (название работы, ФИО, группа, дата).
2. **Цель и задачи** работы.
3. **Описание данных** (источник, переменные, размер выборки).
4. **Этапы анализа** с пояснениями и скриншотами/кодами.
5. **Результаты**:

- таблица коэффициентов;
- значение $R2$ и его интерпретация;
- графики (рассеяния, остатки, QQ-plot).

6. **Выводы**:
   - какие переменные значимы;
   - качество модели;
   - возможные улучшения (например, добавление переменных, трансформация данных).

---

**Контрольные вопросы**

1. Что показывает коэффициент детерминации $R2$?
2. Как интерпретировать $p$-значение коэффициента регрессии?
3. Какие предпосылки должны выполняться для линейной регрессии?
4. В чём разница между обучающей и тестовой выборками?
5. Как проверить наличие гетероскедастичности остатков?

---

**Примечание:** Для выполнения работы можно использовать любой набор данных. Примеры тем:

- прогнозирование цены квартиры по площади и локации;
- анализ влияния рекламы на продажи;
- зависимость урожайности от погодных условий.
-

***Оформление результатов работы***

- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №6. Применение кластеризации для сегментации данных.

**Цель работы:** освоить методику кластерного анализа для сегментации данных: научиться применять алгоритм *k-means*, определять оптимальное число кластеров, интерпретировать результаты и визуализировать их. Кластеризация помогает сегментировать клиентскую базу по покупательскому поведению (частота покупок, средний чек, категории товаров).

***Задачи:***

- Систематизировать подходы к изучению предмета.

• Научить правильному плану работы ИИ

• Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая справка**

**Кластеризация** — метод неконтролируемого машинного обучения, позволяющий разделить набор объектов на группы (кластеры) так, чтобы объекты внутри одного кластера были более похожи друг на друга, чем на объекты из других кластеров.

**Алгоритм *k-means***:

1. Выбрать число кластеров $k$.
2. Инициализировать $k$ центроидов (центров кластеров) — обычно случайным образом.
3. Назначить каждый объект к ближайшему центроиду (по евклидову расстоянию).
4. Пересчитать центроиды как среднее арифметическое всех объектов в кластере.
5. Повторять шаги 3–4 до сходимости (когда центроиды перестают существенно меняться).

**Евклидово расстояние** между точками $A(x1,y1)$ и $B(x2,y2)$:

$d=(x2-x1)2+(y2-y1)2$

**Метод «локтя»** для выбора $k$:

- Запустить *k-means* для $k=1,2,\ldots,K\max$.
- Для каждого $k$ вычислить **внутрикластерную дисперсию** (сумму квадратов расстояний от точек до центроидов).
- Построить график: по оси $X$ — $k$, по оси $Y$ — дисперсия.
- Выбрать $k$ в точке «локтя» (где снижение дисперсии замедляется).

**Ход работы**

**Шаг 1. Подготовка данных**

1. Выберите набор данных (например, Iris из scikit-learn или свой CSV-файл).
2. Загрузите данные и изучите их структуру:

```python
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head())
print(df.info())
```

3. Обработайте пропуски (если есть) и удалите нерелевантные столбцы.
4. Нормализуйте признаки (чтобы они имели одинаковый масштаб):

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

**Шаг 2. Выбор числа кластеров (*k*) методом «локтя»**

1. Запустите *k-means* для разных *k* (например, от 1 до 10):

```python
from sklearn.cluster import KMeans
inertias = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
```

2. Постройте график:

```python
import matplotlib.pyplot as plt
plt.plot(range(1, 11), inertias, 'bo-')
plt.xlabel('Число кластеров (k)')
plt.ylabel('Внутрикластерная дисперсия (inertia)')
plt.title('Метод локтя')
plt.show()
```

3. Определите оптимальное *k* (точка «локтя»).

**Шаг 3. Кластеризация с выбранным *k***

1. Обучите модель:

```python
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
labels = kmeans.fit_predict(X_scaled)
```

2. Добавьте метки кластеров в датасет:

```python
df['Cluster'] = labels
```

**Шаг 4. Визуализация результатов**

1. Если признаков > 2, используйте PCA для снижения размерности:

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

2. Постройте график кластеров:

```python
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            c='red', marker='x', s=200, label='Центроиды')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.title('Результаты кластеризации')
plt.legend()
```

```
plt.show()
```

**Шаг 5. Интерпретация результатов**

1. Проанализируйте состав каждого кластера (средние значения признаков).
2. Опишите, чем отличаются кластеры друг от друга.
3. Предложите возможное применение сегментов (например, для таргетированной рекламы).

**Пример вывода**

Оптимальное число кластеров: $k$=3 (определено методом «локтя»).
Кластер 0: высокие значения признака A, низкие — признака B.
Кластер 1: средние значения обоих признаков.
Кластер 2: низкие значения признака A, высокие — признака B.
Применение: персонализация предложений для каждой группы клиентов.

**Требования к отчёту**

1. Титульный лист (название работы, ФИО, группа).
2. Цель и задачи.
3. Описание данных (источник, признаки, предобработка).
4. График метода «локтя» и обоснование выбора $k$.
5. Визуализация кластеров.
6. Интерпретация результатов (таблица средних по кластерам, выводы).
7. Код (в приложении или встраиванием в текст).

**Контрольные вопросы**

1. В чём отличие кластеризации от классификации?
2. Почему важно нормализовать данные перед *k-means*?
3. Какие ещё метрики расстояния можно использовать в кластеризации?
4. Назовите недостатки алгоритма *k-means*.
5. Как оценить качество кластеризации без истинных меток?

- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №7. Оценка качества модели с использованием ROC-кривой и F-меры.

**Цель работы**: освоить методы оценки качества бинарной классификационной модели через:

- построение ROC-кривой;
- расчёт площади под ROC-кривой (AUC);
- вычисление F-меры ($F_1$-score).

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ

• Показать основные приемы эффективного использования возможностей ИИ

***Время на выполнение работы*:** 90 мин

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретические основы**

**1. Основные метрики классификации**

Для оценки модели используем **матрицу ошибок** (confusion matrix) с четырьмя категориями:

- **TP** (True Positive) — верно классифицированные положительные случаи;
- **TN** (True Negative) — верно классифицированные отрицательные случаи;
- **FP** (False Positive) — ложноположительные срабатывания (ошибка I рода);
- **FN** (False Negative) — ложноотрицательные срабатывания (ошибка II рода).

На их основе вычисляются ключевые метрики:

- **Точность (Precision)**:

Precision=$TP+FPTP$

- **Полнота (Recall, TPR)**:

Recall=TPR=$TP+FNTP$

- **FPR (False Positive Rate)**:

FPR=$FP+TNFP$

**2. ROC-кривая и AUC**

**ROC-кривая** (Receiver Operating Characteristic) — график зависимости TPR от FPR при варьировании порога классификации.

**AUC** (Area Under the Curve) — площадь под ROC-кривой:

- AUC=1 — идеальная модель;
- AUC=0,5 — модель не лучше случайного угадывания;
- AUC<0,5 — модель работает хуже случайного.

**3. F-мера ($F_1$-score)**

Гармоническое среднее точности и полноты:

F1=Precision+Recall2·Precision·Recall

$F_1$-мера особенно полезна при **дисбалансе классов**, так как учитывает оба типа ошибок.

**Порядок выполнения работы**

**Шаг 1. Подготовка данных**

1. Загрузите датасет для бинарной классификации.
2. Разделите данные на обучающую и тестовую выборки (например, 70 % / 3 gef 30 %).
3. Обучите классификационную модель (логистическая регрессия, Random Forest и т. п.).

**Шаг 2. Получение предсказаний**

1. Получите **вероятности** принадлежности к классу 1 на тестовой выборке.
2. Для расчёта метрик используйте **пороги** от 0 до 1 с шагом 0,1.

**Шаг 3. Расчёт TPR и FPR для каждого порога**

Для каждого порога $\tau$:

1. Преобразуйте вероятности в бинарные предсказания:

$y\hat{}i=\{1,0,\text{если } pi\geq\tau\text{иначе}$

2. Постройте матрицу ошибок и вычислите TP, FP, TN, FN.
3. Рассчитайте TPR и FPR по формулам выше.

**Шаг 4. Построение ROC-кривой**

1. Отложите по оси X — FPR, по оси Y — TPR.
2. Соедините точки плавной линией.
3. Добавьте диагональную линию (FPR = TPR) для сравнения с случайным классификатором.

**Шаг 5. Расчёт AUC**

Используйте численное интегрирование (например, метод трапеций) или встроенную функцию (sklearn.metrics.roc_auc_score).

**Шаг 6. Расчёт $F_1$-меры**

1. Для каждого порога вычислите Precision и Recall.
2. Рассчитайте $F_1$-score по формуле.
3. Найдите порог, максимизирующий $F_1$.

**Пример расчёта (упрощённый)**

Пусть для порога $\tau$=0,5 матрица ошибок:

| | Предсказано 0 | Предсказано 1 |
|---|---|---|
| Истинное 0 | TN = 40 | FP = 10 |
| Истинное 1 | FN = 5 | TP = 45 |

Тогда:

- Precision = 45+1045=0,818
- Recall = TPR = 45+545=0,9
- FPR = 10+4010=0,2
- $F_1$ = 0,818+0,92·0,818·0,9≈0,857

**Анализ результатов**

1. **ROC-кривая**:
   - Чем ближе к верхнему левому углу — лучше модель.
   - Диагональ — базовый уровень (случайный классификатор).
2. **AUC**:
   - 0,9 — отличное качество;
   - 0,8–0,9 — хорошее;
   - 0,7–0,8 — удовлетворительное;
   - < 0,7 — низкое.
3. **$F_1$-мера**:
   - Максимизируйте $F_1$ для выбора оптимального порога.
   - Сравните $F_1$ с Accuracy для оценки влияния дисбаланса классов.

**Требования к отчёту**

1. Описание датасета и модели.
2. Таблица с TPR и FPR для разных порогов.
3. График ROC-кривой с подписью AUC.
4. График $F_1$-меры в зависимости от порога.
5. Выводы:
   - Какое значение AUC получено?
   - Какой порог оптимален по $F_1$?
   - Как модель справляется с дисбалансом классов?

**Контрольные вопросы**

1. Почему AUC инвариантен к порогу классификации?
2. В каких случаях $F_1$-мера предпочтительнее Accuracy?
3. Как интерпретировать ROC-кривую, если она ниже диагонали?
4. Как AUC ведёт себя при сильном дисбалансе классов?
5. Почему $F_1$-мера использует гармоническое среднее, а не арифметическое?

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №8. Настройка гиперпараметров модели с использованием GridSearchCV.

**Цель работы** освоить методику настройки гиперпараметров модели машинного обучения с помощью инструмента GridSearchCV из библиотеки scikit-learn. Научиться подбирать оптимальные комбинации гиперпараметров для повышения качества модели.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретические основы**

**Гиперпараметры** — это параметры модели, которые задаются *до начала обучения* и не обновляются в процессе оптимизации (в отличие от параметров модели). Примеры:

- для SVM: $C$, $\gamma$, тип ядра;
- для случайного леса: n_estimators, max_depth, min_samples_leaf;
- для KNN: n_neighbors, weights, p.

**GridSearchCV** — метод полного перебора всех комбинаций заданных гиперпараметров с кросс-валидацией. Для каждой комбинации:

1. Модель обучается на $k$ частях данных (кросс-валидация).
2. Оценивается средняя метрика качества (например, точность).
3. Выбирается комбинация с наилучшим средним результатом.

**Инструменты**

- Python 3.x
- Библиотеки: scikit-learn, numpy, pandas

**Ход работы**

**Шаг 1. Импорт библиотек и загрузка данных**

```
python
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Загрузим датасет (пример — Iris):

```
python
iris = load_iris()
X, y = iris.data, iris.target

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

**Шаг 2. Определение модели и сетки параметров**

**Пример для SVM:**

```
python
svc = SVC()

param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [0.1, 0.01, 0.001],
    'kernel': ['linear', 'rbf']
}
```

**Пример для случайного леса:**

```
python
rf = RandomForestClassifier(random_state=42)

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_leaf': [1, 2, 4]
}
```

**Шаг 3. Настройка** `GridSearchCV`

```
python
grid_search = GridSearchCV(
    estimator=svc,          # модель
    param_grid=param_grid,  # сетка параметров
    cv=5,                   # 5-кратная кросс-валидация
    scoring='accuracy',     # метрика качества
    n_jobs=-1,              # параллельные вычисления
    verbose=1               # вывод прогресса
)
```

**Шаг 4. Обучение и поиск оптимальных параметров**

```
python
grid_search.fit(X_train, y_train)
```

**Шаг 5. Анализ результатов**

```python
print("Лучшие гиперпараметры:", grid_search.best_params_)
print("Лучшая оценка (кросс-валидация):", grid_search.best_score_)

# Оценка на тестовой выборке
best_model = grid_search.best_estimator_
test_score = best_model.score(X_test, y_test)
print("Оценка на тесте:", test_score)
```

**Шаг 6. Визуализация результатов (опционально)**

Можно вывести таблицу всех комбинаций и их оценок:

```python
results = pd.DataFrame(grid_search.cv_results_)
print(results[['param_C', 'param_gamma', 'param_kernel', 'mean_test_score']])
```

**Пример полного кода (SVM + Iris)**

```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Загрузка данных
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Настройка GridSearchCV
svc = SVC()
param_grid = {'C': [0.1, 1, 10], 'gamma': [0.01, 0.1], 'kernel': ['rbf']}
grid_search = GridSearchCV(svc, param_grid, cv=5, scoring='accuracy')

# Обучение
grid_search.fit(X_train, y_train)

# Результаты
print("Лучшие параметры:", grid_search.best_params_)
print("Лучшая кросс-валидация:", grid_search.best_score_)
print("Тест:", grid_search.best_estimator_.score(X_test, y_test))
```

**Контрольные вопросы**

1. Что такое гиперпараметры? Приведите 3 примера для разных моделей.
2. В чём отличие `GridSearchCV` от `RandomizedSearchCV`?
3. Почему важно использовать кросс-валидацию при подборе гиперпараметров?
4. Как интерпретировать `best_score_` из `GridSearchCV`?
5. Какие недостатки у метода полного перебора (`GridSearchCV`)?

**Требования к отчёту**

1. Цель работы.
2. Краткое описание метода `GridSearchCV`.
3. Код с комментариями (шаги 1–6).
4. Результаты подбора гиперпараметров (лучшие значения и оценки).
5. Ответы на контрольные вопросы.
6. Выводы (как подбор гиперпараметров повлиял на качество модели).

**Дополнительные задания (по желанию)**

1. Сравните результаты `GridSearchCV` и `RandomizedSearchCV` для той же модели.
2. Подберите гиперпараметры для другой модели (например, `RandomForestClassifier`).
3. Используйте другую метрику качества (например, `f1_macro`).

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы. (устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №9. Реализация многослойного перцептрона (MLP) для задачи классификации.

**Цель работы**: реализовать и обучить многослойный перцептрон (MLP) для решения задачи классификации, исследовать влияние гиперпараметров на качество модели.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая часть**

**Многослойный перцептрон** (Multilayer Perceptron, MLP) — это полносвязная нейронная сеть прямого распространения, состоящая из:

- входного слоя;
- одного или нескольких скрытых слоёв;
- выходного слоя.

**Основные компоненты**:

1. **Нейроны** — вычислительные единицы, выполняющие нелинейное преобразование входных данных.
2. **Веса связей** ($W$) — параметры, определяющие силу связи между нейронами.
3. **Смещение** ($b$) — дополнительный параметр для сдвига активации.
4. **Функция активации** — нелинейное преобразование выхода нейрона.

**Типичные функции активации**:

- сигмоида: $\sigma(z)=1+e-z1$;
- ReLU: $f(z)=\max(0,z)$;
- tanh: $\tanh(z)=ez+e-zez-e-z$.

**Функция потерь** для классификации:

- категориальная кросс-энтропия:

$L=-\sum i=1Cyi\log(y\hat{}i)$,

где $C$ — число классов, $yi$ — истинная метка, $y\hat{}i$ — предсказание.

**Алгоритм обучения** — обратное распространение ошибки (backpropagation) с градиентным спуском.

**Практическая реализация**

**Шаг 1. Импорт библиотек**

```
python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
```

**Шаг 2. Генерация данных**

```
python
# Создаём синтетический набор данных для классификации
X, y = make_classification(
    n_samples=1000,
    n_features=2,
    n_classes=2,
```

```
    n_redundant=0,
    random_state=42
)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Нормализация данных
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

**Шаг 3. Реализация MLP**

python

```
class MLP:
    def __init__(self, input_size, hidden_sizes, output_size, lr=0.01):
        self.layers = []
        self.lr = lr

        # Инициализация весов
        sizes = [input_size] + hidden_sizes + [output_size]
        for i in range(len(sizes) - 1):
            W = np.random.randn(sizes[i], sizes[i+1]) * 0.5
            b = np.zeros((1, sizes[i+1]))
            self.layers.append({'W': W, 'b': b})

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-np.clip(x, -500, 500)))

    def sigmoid_derivative(self, x):
        return x * (1 - x)

    def forward(self, X):
        activations = [X]
        for layer in self.layers:
            z = np.dot(activations[-1], layer['W']) + layer['b']
            a = self.sigmoid(z)
            activations.append(a)
        return activations

    def backward(self, activations, y):
        m = y.shape[0]
        dA = activations[-1] - y  # для бинарной классификации
```

```
        for i in reversed(range(len(self.layers))):
            dZ = dA * self.sigmoid_derivative(activations[i+1])
            dW = np.dot(activations[i].T, dZ) / m
            db = np.sum(dZ, axis=0, keepdims=True) / m

            if i > 0:
                dA = np.dot(dZ, self.layers[i]['W'].T)

            self.layers[i]['W'] -= self.lr * dW
            self.layers[i]['b'] -= self.lr * db

    def train(self, X, y, epochs):
        y = y.reshape(-1, 1)
        losses = []

        for epoch in range(epochs):
            activations = self.forward(X)
            loss = np.mean(-(y * np.log(activations[-1]) + (1-y) * np.log(1-activations[-1])))
            losses.append(loss)

            self.backward(activations, y)

            if epoch % 100 == 0:
                print(f'Epoch {epoch}, Loss: {loss:.4f}')

        return losses

    def predict(self, X):
        activations = self.forward(X)
        return (activations[-1] > 0.5).astype(int).flatten()
```

**Шаг 4. Обучение модели**

python

```
# Создание и обучение MLP
mlp = MLP(input_size=2, hidden_sizes=[4, 3], output_size=1, lr=0.1)
losses = mlp.train(X_train, y_train, epochs=1000)

# Предсказание на тестовой выборке
y_pred = mlp.predict(X_test)
```

**Шаг 5. Оценка качества**

python

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Точность: {accuracy:.4f}')
print('Отчёт о классификации:')
print(classification_report(y_test, y_pred))
```

```
# Визуализация потерь
plt.plot(losses)
plt.title('Функция потерь')
plt.xlabel('Эпоха')
plt.ylabel('Потеря')
plt.show()
```

**Анализ результатов**

1. **Точность модели** (accuracy) — доля правильно классифицированных примеров.
2. **Кривая потерь** — показывает сходимость алгоритма обучения.
3. **Отчёт классификации** — включает precision, recall, F1-score для каждого класса.

**Исследование гиперпараметров**

Проведите эксперименты с варьированием:

- числа скрытых слоёв;
- количества нейронов в слоях;
- скорости обучения (`lr`);
- функции активации.

Пример исследования:

```
python
hidden_configs = [[2], [4], [4, 2], [6, 4, 2]]
accuracies = []

for config in hidden_configs:
    mlp = MLP(2, config, 1, lr=0.1)
    mlp.train(X_train, y_train, 500)
    y_pred = mlp.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)
    print(f'Структура {config}: Точность = {acc:.4f}')
```

Постройте график зависимости точности от конфигурации сети.

**Выводы**

В ходе работы:

1. Реализован MLP с возможностью настройки архитектуры.
2. Проведено обучение на синтетическом наборе данных.
3. Выполнена оценка качества классификации.
4. Исследовано влияние гиперпараметров на результат.

**Ключевые наблюдения**:

- Увеличение числа слоёв/нейронов может привести к переобучению.
- Оптимальная скорость обучения обеспечивает быструю сходимость без осцилляций.
- Нормализация данных существенно улучшает обучение.

**Дополнительные задания (по желанию)**

1. Добавить другие функции активации (ReLU, tanh).
2. Реализовать регуляризацию (L1/L2).

3. Применить MLP к реальному датасету (например, Iris или MNIST).
4. Добавить визуализацию границ решения для двумерного случая.

- ***Оформление результатов работы***

- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №10. Создание сверточной нейронной сети для распознавания изображений.

**Цель работы**: освоить процесс проектирования, обучения и тестирования свёрточной нейронной сети (CNN) для задачи классификации изображений с использованием фреймворка Keras/TensorFlow.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Задачи**

- подготовить датасет для обучения;

- построить архитектуру CNN;
- обучить модель на тренировочных данных;
- оценить качество модели на тестовой выборке;
- проанализировать результаты и сделать выводы.

**Теоретическая справка**

**Свёрточная нейронная сеть (CNN)** — специализированный тип нейронных сетей для обработки изображений. Основные блоки:

- **Свёрточные слои** (`Conv2D`) — извлекают локальные признаки с помощью фильтров.
- **Слои пулинга** (`MaxPooling2D`) — уменьшают размерность карт признаков.
- **Полносвязные слои** (`Dense`) — выполняют классификацию на основе выученных признаков.
- **Функции активации** (ReLU, Softmax) — вносят нелинейность.
- **Регуляризация** (`Dropout`, `BatchNormalization`) — предотвращает переобучение.

**Инструменты**

- Python 3.x
- Библиотеки: `tensorflow`, `keras`, `numpy`, `matplotlib`, `sklearn`
- Среда: Jupyter Notebook / Google Colab / локальная IDE
- **. Ход работы**

**Шаг 1. Импорт библиотек**

```
python
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

**Шаг 2. Загрузка и предобработка данных**

Пример для датасета CIFAR-10:

```
python
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

# Нормализация пикселей [0, 255] → [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Разделение тренировочной выборки (80% / 20%)
x_train, x_val, y_train, y_val = train_test_split(
    x_train, y_train, test_size=0.2, random_state=42
)
```

**Шаг 3. Построение архитектуры CNN**

```
python
model = models.Sequential([
```

```
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')  # 10 классов CIFAR-10
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

**Шаг 4. Обучение модели**

```
python
history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=32,
    validation_data=(x_val, y_val)
)
```

**Шаг 5. Оценка качества**

```
python
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"Точность на тестовой выборке: {test_acc:.4f}")
```

**Шаг 6. Визуализация результатов**

```
python
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Точность (обучение)')
plt.plot(history.history['val_accuracy'], label='Точность (валидация)')
plt.title('Точность модели')
plt.xlabel('Эпоха')
plt.ylabel('Точность')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Ошибка (обучение)')
```

```
plt.plot(history.history['val_loss'], label='Ошибка (валидация)')
plt.title('Функция потерь')
plt.xlabel('Эпоха')
plt.ylabel('Ошибка')
plt.legend()

plt.show()
```

**Анализ результатов**

1. **Точность (accuracy)** — доля правильно классифицированных изображений.
2. **Функция потерь (loss)** — мера расхождения предсказаний с истинными метками.
3. **Переобучение** — если точность на валидации падает, а на обучении растёт.
4. **Недообучение** — низкая точность на обеих выборках.

**Возможные улучшения**

- Увеличение глубины сети (больше свёрточных слоёв).
- Использование предобученных моделей (Transfer Learning: VGG16, ResNet).
- Аугментация данных (поворот, сдвиг, изменение яркости).
- Настройка гиперпараметров (скорость обучения, размер батча).

**Требования к отчёту**

1. **Титульный лист** (ФИО, группа, дата).
2. **Цель и задачи** работы.
3. **Описание датасета** (размер, классы, примеры изображений).
4. **Архитектура модели** (схема или код).
5. **Графики обучения** (точность и ошибка).
6. **Результаты тестирования** (точность на тесте, матрица ошибок).
7. **Выводы** (анализ работы модели, предложения по улучшению).
8. **Полный код** в приложении.

**Контрольные вопросы**

1. В чём отличие свёрточных слоёв от полносвязных?
2. Зачем нужен слой пулинга?
3. Как предотвратить переобучение в CNN?
4. Что такое функция активации ReLU и почему она популярна в CNN?
5. Как интерпретировать график функции потерь при обучении?

**Дополнительные задания (по желанию)**

1. Реализовать аугментацию данных с помощью `ImageDataGenerator`.
2. Сравнить точность CNN с полносвязной сетью на том же датасете.
3. Протестировать модель на собственных изображениях (загруженных извне).

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и

кодированию.

- **Ответить на контрольные вопросы.(устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №11. Реализация рекуррентной нейронной сети для анализа временных рядов.

**Цель работы**; реализовать рекуррентную нейронную сеть (RNN) для прогнозирования значений временного ряда, изучить особенности её работы и оценить качество предсказаний.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1.Методические рекомендации для практических занятий

2.Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая часть**

**Временной ряд** — последовательность данных, упорядоченная по времени. Примеры: курс валют, температура воздуха, продажи товара.

**Рекуррентные нейронные сети (RNN)** — класс нейронных сетей, специально разработанный для работы с последовательными данными. Их ключевая особенность — наличие «памяти»: на каждом шаге сеть получает не только новый входной сигнал, но и информацию о предыдущих состояниях.

**LSTM (Long Short-Term Memory)** — улучшенная архитектура RNN, способная улавливать долгосрочные зависимости в данных благодаря специальным «ячейкам памяти».

**Основные компоненты LSTM-ячейки:**

- **Забывающий вентиль** (*ft*): решает, какую информацию сохранить/удалить из состояния.

- **Входной вентиль** (*it*): определяет, какие новые данные добавить.
- **Выходной вентиль** (*ot*): контролирует, что передать на выход.
- **Состояние ячейки** (*Ct*): «память» сети.

Математически это выражается формулами:

*ftitC~tCtotht=σ(Wf·[ht−1,xt]+bf)=σ(Wi·[ht−1,xt]+bi)=tanh(WC·[ht−1,xt]+bC)=ft·Ct−1+it ·C~t=σ(Wo·[ht−1,xt]+bo)=ot·tanh(Ct)*

где:

- *xt* — входной вектор на шаге *t*;
- *ht*−1 — скрытое состояние предыдущего шага;
- *W* и *b* — веса и смещения вентилей;
- *σ* — сигмоида;
- tanh — гиперболический тангенс.

**Практическая часть**

**Шаг 1. Подготовка окружения**

Установите необходимые библиотеки:

```
python
pip install numpy pandas matplotlib scikit-learn tensorflow
```

**Шаг 2. Загрузка и предобработка данных**

1. Загрузите временной ряд (пример — курс акций):

```
python
import pandas as pd
data = pd.read_csv('stock_prices.csv')
prices = data['Close'].values.reshape(-1, 1)
```

2. Нормализуйте данные (диапазон [0, 1]):

```
python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(prices)
```

3. Разделите на обучающую и тестовую выборки (например, 80 % / 20 %):

```
python
train_size = int(len(scaled_data) * 0.8)
train_data = scaled_data[:train_size]
test_data = scaled_data[train_size:]
```

**Шаг 3. Создание последовательностей**

Преобразуйте данные в последовательности фиксированной длины (например, 60 дней):

```
python
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 60
```

```
X_train, y_train = create_sequences(train_data, seq_length)
X_test, y_test = create_sequences(test_data, seq_length)
```

**Шаг 4. Построение модели LSTM**

```
python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(seq_length, 1)),
    LSTM(50, return_sequences=False),
    Dense(25),
    Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')
```

**Параметры модели:**

- Два слоя LSTM с 50 нейронами каждый.
- Первый слой возвращает последовательность (`return_sequences=True`), второй — только последнее состояние.
- Полносвязные слои (Dense) для финального прогноза.

**Шаг 5. Обучение модели**

```
python
history = model.fit(
    X_train, y_train,
    batch_size=32,
    epochs=50,
    validation_data=(X_test, y_test),
    verbose=1
)
```

**Шаг 6. Прогнозирование и оценка**

1. Получите предсказания:

```
python
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)  # Обратное масштабирование
```

2. Оцените качество (MSE, MAE):

```
python
from sklearn.metrics import mean_squared_error, mean_absolute_error

mse = mean_squared_error(y_test, predictions)
mae = mean_absolute_error(y_test, predictions)
print(f'MSE: {mse:.2f}, MAE: {mae:.2f}')
```

3. Визуализируйте результаты:

```
python
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(14, 5))
plt.plot(y_test, label='Фактические значения')
plt.plot(predictions, label='Прогноз')
plt.legend()
plt.show()
```

**Анализ результатов**

1. **Ошибка (MSE/MAE)**: чем меньше — тем лучше. Сравните с базовыми моделями (например, скользящим средним).
2. **Визуализация**: оцените, насколько прогноз повторяет тренды исходного ряда.
3. **Долгосрочность**: проверьте, как ухудшается точность при увеличении горизонта прогноза.

**Возможные улучшения**

1. **Настройка гиперпараметров**:
   - количество слоёв LSTM;
   - число нейронов;
   - длина последовательности (`seq_length`);
   - скорость обучения (`learning_rate`).
2. **Добавление слоёв**:
   - Dropout для борьбы с переобучением;
   - Bidirectional LSTM для учёта контекста в обе стороны.
3. **Другие архитектуры**:
   - GRU (упрощённая альтернатива LSTM);
   - Transformer для длинных последовательностей.
4. **Дополнительные признаки**:
   - технические индикаторы (RSI, MACD);
   - внешние данные (новости, макроэкономические показатели).

**Контрольные вопросы**

1. В чём ключевое отличие RNN от полносвязных сетей?
2. Почему LSTM лучше справляется с долгосрочными зависимостями, чем классическая RNN?
3. Зачем нужна нормализация данных перед обучением?
4. Как интерпретировать значения MSE и MAE?
5. Какие ещё метрики можно использовать для оценки качества прогноза?

**Вывод**

В ходе работы вы:

- изучили принцип работы RNN и LSTM;
- реализовали модель для прогнозирования временного ряда;
- оценили качество предсказаний и выявили возможные пути улучшения.

**Итог**: RNN/LSTM — эффективный инструмент для анализа последовательностей, но требует тщательной настройки и достаточного объёма данных.

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы. (устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №12. Проектирование архитектуры ИИ-системы с учетом модульности и масштабируемости.

**Цель работы** — научиться разрабатывать модульные ИИ-системы, соответствующие требованиям производительности и безопасности, а также внедрять разработанные ИИ-модули в комплексные программные системы.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

Сформировать навыки проектирования архитектуры ИИ-системы, обеспечивающей:

- **модульность** — возможность независимой разработки, тестирования и замены компонентов;
- **масштабируемость** — способность эффективно наращивать производительность при росте нагрузки.

**Исходные данные**

Предположим, разрабатывается ИИ-система для **аналитики клиентского поведения в e-commerce** (прогнозирование оттока, персонализация рекомендаций, выявление аномалий в покупках).

**Шаг 1. Определение ключевых функциональных модулей**

Выделим основные блоки системы:

1. **Модуль сбора данных**
   - получает потоки данных из CRM, веб-аналитики, мобильных приложений;
   - обеспечивает первичную валидацию и буферизацию.
2. **Модуль предобработки**
   - очищает данные (удаление дублей, заполнение пропусков);
   - трансформирует признаки (нормализация, кодирование категорий);
   - формирует витрины для моделей.
3. **Модуль моделей машинного обучения**
   - содержит независимые модели для разных задач (классификация оттока, рекомендательный движок);
   - поддерживает версионирование моделей.
4. **Модуль инференса**
   - принимает запросы от внешних систем;
   - вызывает нужные модели;
   - возвращает результаты в реальном времени.
5. **Модуль мониторинга**
   - отслеживает качество моделей (drift, метрики);
   - логирует ошибки и задержки;
   - отправляет алерты.
6. **API-шлюз**
   - единый вход для внешних запросов;
   - аутентификация, лимитирование запросов.

**Шаг 2. Архитектурная схема (описание)**

Система строится по принципу **микросервисной архитектуры**:

- Каждый модуль — отдельный сервис с собственным API.
- Взаимодействие через **асинхронные сообщения** (Kafka/RabbitMQ) для потоковых данных и **синхронные HTTP/gRPC** для инференса.
- Данные хранятся в:
  - **хранилище сырых данных** (S3/HDFS);
  - **аналитическое хранилище** (ClickHouse/BigQuery);
  - **кэш результатов** (Redis).
- Оркестрация контейнеров — Kubernetes.

**Шаг 3. Обеспечение модульности**

Для каждого модуля соблюдаются принципы:

1. **Слабая связанность**
   - сервисы общаются через чётко определённые интерфейсы (OpenAPI, Protobuf);
   - изменения в одном модуле не ломают другие.
2. **Независимое развёртывание**
   - каждый сервис имеет свой Docker-образ и Helm-чарты;
   - обновления проводятся без простоя всей системы.
3. **Изоляция данных**
   - модули не имеют прямого доступа к БД соседей;
   - обмен данными — через API или шины событий.
4. **Стандартизированные интерфейсы**
   - все модели принимают входные данные в формате $\text{JSON}$ с заданной схемой;

- ответы — единообразный $\text{JSON}$ с полями prediction, confidence, model_version.

**Шаг 4. Обеспечение масштабируемости**

Применяются следующие техники:

1. **Горизонтальное масштабирование**
   - Kubernetes автоматически добавляет реплики сервисов при росте нагрузки;
   - балансировка запросов (LoadBalancer, Ingress).
2. **Шардирование данных**
   - аналитическое хранилище разбито на шарды по времени/пользователям;
   - кэш Redis использует кластерный режим.
3. **Асинхронная обработка**
   - потоковые данные накапливаются в Kafka, обрабатываются батчами;
   - длительные задачи (переобучение моделей) — в очереди (Celery/Airflow).
4. **Кэширование**
   - результаты частых запросов хранятся в Redis;
   - TTL для актуальности данных.
5. **Автоматическое управление ресурсами**
   - HPA (Horizontal Pod Autoscaler) в Kubernetes;
   - лимиты CPU/RAM для предотвращения «расползания» сервисов.

**Шаг 5. Пример взаимодействия модулей**

**Сценарий: прогноз оттока клиента**

1. API-шлюз принимает запрос $\text{POST /predict-churn}$ с $\text{user_id}$.
2. Модуль инференса запрашивает признаки пользователя из аналитического хранилища.
3. Модель классификации оттока (версия 2.1) делает предсказание.
4. Результат возвращается через API-шлюз.
5. Событие логируется в Kafka для мониторинга.

**Шаг 6. Инструменты и технологии**

- **Оркестрация**: Kubernetes, Helm.
- **Обмен сообщениями**: Apache Kafka.
- **Хранилища**: S3, ClickHouse, Redis.
- **Модели**: Python (scikit-learn, TensorFlow), MLflow для версионирования.
- **API**: FastAPI (Python), gRPC.
- **Мониторинг**: Prometheus, Grafana, ELK.

**Шаг 7. Критерии оценки архитектуры**

1. **Модульность**
   - можно заменить модель рекомендаций, не трогая модуль сбора данных;
   - добавление нового типа данных требует изменения только модуля предобработки.
2. **Масштабируемость**
   - при росте числа запросов в 10 раз система добавляет реплики сервисов автоматически;
   - задержка ответа не превышает 200 мс при нагрузке до 1000 RPS.
3. **Отказоустойчивость**
   - реплики сервисов обеспечивают работу при падении узла;
   - данные сохраняются в распределённом хранилище.

**Вывод**

В ходе работы спроектирована архитектура ИИ-системы, отвечающая требованиям:

- **Модульность** достигнута за счёт микросервисного подхода, стандартизированных интерфейсов и изоляции данных.

- **Масштабируемость** обеспечена оркестрацией Kubernetes, асинхронной обработкой и шардированием.

Такая архитектура позволяет:

- быстро внедрять новые модели;
- наращивать мощность без переписания кода;
- минимизировать простои при обновлениях.
- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы. (устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №13. Контейнеризация ИИ-модели с использованием Docker.

**Цель работы:** освоить процесс контейнеризации ИИ-модели с помощью Docker: создать Docker-образ, запустить контейнер, проверить работоспособность модели в изолированной среде.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Теоретическая часть**

**Docker** — платформа для автоматизации развёртывания и управления приложениями в контейнерах. Контейнер включает:

- исполняемый код приложения;
- среду выполнения;
- системные инструменты;
- библиотеки и настройки.

**Преимущества для ИИ-моделей:**

- консистентность окружения на разных платформах;
- изоляция зависимостей;
- простота масштабирования;
- воспроизводимость результатов.

**Ключевые компоненты:**

- **Dockerfile** — скрипт для сборки образа;
- **образ (image)** — шаблон контейнера;
- **контейнер (container)** — запущенный экземпляр образа;
- **registry** — хранилище образов (например, Docker Hub).

**Практическая часть**

**Шаг 1. Подготовка проекта**

Создайте структуру файлов:

```
project/
├── app/
│   ├── model.pkl          # обученная модель
│   └── predict.py       # код для инференса
├── requirements.txt       # зависимости
├── Dockerfile            # инструкция для сборки
└── .dockerignore          # игнорируемые файлы
```

**Шаг 2. Написание requirements.txt**

Пример содержимого:

```
txt
fastapi==0.115.0
uvicorn==0.30.2
scikit-learn==1.5.2
numpy==1.26.4
pydantic==2.9.2
```

**Шаг 3. Создание Dockerfile**

Пример Dockerfile:

```
dockerfile
# Базовый образ с Python
FROM python:3.11-slim

# Метаданные
LABEL maintainer="your-email@example.com"
```

```
# Установка рабочей директории
WORKDIR /app

# Копирование зависимостей
COPY requirements.txt .

# Установка пакетов (без кэша для уменьшения размера образа)
RUN pip install --no-cache-dir -r requirements.txt

# Копирование кода приложения
COPY app/ .

# Открытие порта для API
EXPOSE 8000

# Команда запуска приложения
CMD ["uvicorn", "predict:app", "--host", "0.0.0.0", "--port", "8000"]
```

**Шаг 4. Настройка .dockerignore**

Пример .dockerignore:

txt

```
__pycache__/
*.pyc
*.pyo
*.pyd
.git/
.env
Dockerfile
docker-compose.yml
```

**Шаг 5. Код приложения (predict.py)**

Пример простого API на FastAPI:

python

```
from fastapi import FastAPI
import pickle
import numpy as np

# Загрузка модели
with open("model.pkl", "rb") as f:
    model = pickle.load(f)

app = FastAPI()

@app.post("/predict")
def predict(data: list):
    input_data = np.array(data).reshape(1, -1)
    prediction = model.predict(input_data)
```

```
    return {"prediction": prediction.tolist()}
```

**Шаг 6. Сборка Docker-образа**

В терминале выполните:

```
bash
docker build -t ai-model:v1 .
```

Проверка образов:

```
bash
docker images
```

**Шаг 7. Запуск контейнера**

Запуск с пробросом порта:

```
bash
docker run -p 8000:8000 ai-model:v1
```

Параметры:

- `-p 8000:8000` — проброс порта 8000 контейнера на хост;
- `ai-model:v1` — имя образа.

**Шаг 8. Проверка работоспособности**

1. Откройте браузер или используйте `curl`:

```
bash
curl -X POST "http://localhost:8000/predict" \
    -H "Content-Type: application/json" \
    -d "[5.1, 3.5, 1.4, 0.2]"
```

2. Ожидаемый ответ:

```
json
{"prediction": [0]}
```

**Шаг 9. Управление контейнерами**

Основные команды:

```
bash
# Список запущенных контейнеров
docker ps

# Остановка контейнера
docker stop <container-id>

# Удаление контейнера
docker rm <container-id>

# Просмотр логов
docker logs <container-id>
```

**Шаг 10. Публикация образа**

1. Авторизуйтесь в Docker Hub:

```
bash
docker login
```

2. Пометьте образ:

```
bash
```

```
docker tag ai-model:v1 your-username/ai-model:v1
```

3. Загрузите в реестр:

```
bash
docker push your-username/ai-model:v1
```

**Контрольные вопросы**

1. Что такое Docker-контейнер и чем он отличается от виртуальной машины?
2. Для чего нужен файл Dockerfile?
3. Объясните назначение команд `COPY`, `RUN`, `CMD` в Dockerfile.
4. Как пробросить порт контейнера на хост-систему?
5. Какие преимущества даёт контейнеризация для развёртывания ИИ-моделей?

**Отчёт по работе**

Подготовьте отчёт, включающий:

1. Скриншот структуры проекта.
2. Содержимое Dockerfile и requirements.txt.
3. Логи сборки образа (`docker build`).
4. Результат тестирования API (скриншот или вывод `curl`).
5. Ответы на контрольные вопросы.

**Дополнительные задания (по желанию)**

1. Добавьте в Dockerfile создание непривилегированного пользователя.
2. Настройте мультистадийную сборку для уменьшения размера образа.
3. Разверните контейнер с GPU-поддержкой (используйте образ `nvidia/cuda`).
4. Создайте `docker-compose.yml` для запуска модели с базой данных.

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы. (устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Практическая работа №14. Развертывание ИИ-системы в Kubernetes.

**Цель работы**: освоить процесс развёртывания ИИ-приложения в кластере Kubernetes, включая подготовку образа, конфигурирование ресурсов и проверку работоспособности.

***Задачи:***

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ

- Показать основные приемы эффективного использования возможностей ИИ

***Оборудование, технические средства и инструменты:***

Методические рекомендации для практических занятий

Компьютер с подключением к сети Интернет

***Ход практического занятия:***

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

***Теоретические и учебно-методические материалы по теме практической работы***

**Используемые инструменты**

- Kubernetes (кластер)
- Docker
- kubectl (CLI-инструмент для управления Kubernetes)
- Git (для версионирования конфигураций)
- Браузер (для тестирования API)

**Исходное приложение**

В качестве примера используем **ASR Whisper** — модель распознавания речи (преобразование аудио в текст).

**Шаг 1. Подготовка Docker-образа**

1. **Локальный запуск для тестирования**

Выполните команду:

```
bash
docker run -p 9000:9000 \
  -e ASR_MODEL=tiny \
  -e ASR_ENGINE=faster_whisper \
  onerahmet/openai-whisper-asr-webservice:latest
```

2. **Проверка работоспособности**

Откройте в браузере `http://localhost:9000`, выберите метод `POST`, нажмите `Try It Out`.

Параметры:

- `transcribe` — для получения текста;
- `translate` — для перевода.

3. **Создание собственного образа (опционально)**

Для production-контура соберите свой образ на основе Dockerfile, указав:

- версию модели (`tiny`, `base`, `small` и т. д.);
- движок (`faster_whisper` для CPU-оптимизации).

**Шаг 2. Создание Namespace в Kubernetes**

1. **Конфигурационный файл**

Создайте файл `01-namespace.yaml`:

```
yaml
kind: Namespace
apiVersion: v1
metadata:
  name: whisper-ai
```

2. **Применение конфигурации**

```
bash
kubectl apply -f 01-namespace.yaml
```

3. **Проверка**

```
bash
kubectl get ns
```

Убедитесь, что `whisper-ai` присутствует в списке.

**Шаг 3. Развёртывание Deployment**

1. **Файл конфигурации**

Создайте `02-deployment.yaml`:

```
yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: whisper-deployment
  namespace: whisper-ai
spec:
  replicas: 2
  selector:
    matchLabels:
      app: whisper
  template:
    metadata:
      labels:
        app: whisper
    spec:
      containers:
      - name: whisper
        image: onerahmet/openai-whisper-asr-webservice:latest
        ports:
        - containerPort: 9000
```

```
        env:
        - name: ASR_MODEL
          value: "tiny"
        - name: ASR_ENGINE
          value: "faster_whisper"
```

2. **Запуск Deployment**

```
bash
kubectl apply -f 02-deployment.yaml
```

3. **Проверка подов**

```
bash
kubectl get pods -n whisper-ai
```

Статус должен быть `Running`.

**Шаг 4. Настройка Service**

1. **Файл конфигурации**

Создайте `03-service.yaml`:

```
yaml
apiVersion: v1
kind: Service
metadata:
  name: whisper-service
  namespace: whisper-ai
spec:
  type: NodePort
  selector:
    app: whisper
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9000
```

2. **Применение**

```
bash
kubectl apply -f 03-service.yaml
```

3. **Получение внешнего порта**

```
bash
kubectl get service -n whisper-ai
```

Найдите значение в колонке `NODE-PORT` (например, `31000`).

**Шаг 5. Доступ к приложению**

1. **Определение IP-адреса узла**

```
bash
kubectl get nodes -o wide
```

Возьмите IP из колонки `INTERNAL-IP`.

2. **Тестирование API**

Откройте в браузере:

`http://<NODE-IP>:<NODE-PORT>`

Например: `http://192.168.1.100:31000`.

3. **Отправка запроса**

В интерфейсе API выберите:

- метод `POST`;
- параметры (`transcribe` или `translate`);
- загрузите аудиофайл.

**Шаг 6. Мониторинг и оптимизация**

1. **Просмотр логов**

```
bash
kubectl logs -n whisper-ai whisper-deployment-<pod-id> -c whisper
```

2. **Масштабирование**

Увеличьте число реплик:

```
bash
kubectl scale deployment/whisper-deployment -n whisper-ai --replicas=3
```

3. **Настройка лимитов**

Добавьте в `Deployment` секцию `resources`:

```
yaml
resources:
  requests:
    memory: "512Mi"
    cpu: "250m"
  limits:
    memory: "1Gi"
    cpu: "500m"
```

**Шаг 7. Дополнительные улучшения (по желанию)**

1. **Ingress**

Настройте Ingress для доступа по доменному имени.

2. **Helm Chart**

Создайте шаблон для упрощённого деплоя:

```
bash
helm create whisper-chart
```

3. **Мониторинг**

Установите Prometheus + Grafana для отслеживания:

- загрузки CPU/памяти;
- времени ответа API;
- числа ошибок.

4. **Автомасштабирование**

Настройте Horizontal Pod Autoscaler (HPA):

```
bash
kubectl autoscale deployment/whisper-deployment -n whisper-ai \
  --cpu-percent=80 --min=2 --max=5
```

**Контрольные вопросы**

1. Для чего используется `Namespace` в Kubernetes?
2. Чем отличается `Deployment` от `Service`?
3. Как проверить статус подов в определённом namespace?
4. Какие параметры можно задать в `resources` для ограничения потребления ресурсов?
5. Как масштабировать приложение без остановки сервиса?

**Отчёт по работе**

Подготовьте документ, содержащий:

1. Скриншоты:
   - списка подов (`kubectl get pods -n whisper-ai`);
   - конфигурации Service (`kubectl get service -n whisper-ai`);
   - ответа API в браузере.
2. Описание шагов развёртывания.
3. Ответы на контрольные вопросы.
4. Выводы о преимуществах Kubernetes для ИИ-систем.

- ***Оформление результатов работы***
- Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:
- ***Содержание отчёта***
- 1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

- **Ответить на контрольные вопросы. (устно)**
- ***Сформулировать выводы по результатам работы.***
- ***Сдать и защитить работу.***

## Критерии оценки

**Оценка «отлично»** ставится в том случае, если студент:

- свободно применяет полученные знания при выполнении практических заданий;
- выполнил работу в полном объеме с соблюдением необходимой последовательности действий;
- в письменном отчете по работе правильно и аккуратно выполнены все записи;
- при ответах на контрольные вопросы правильно понимает их сущность, дает точное определение и истолкование основных понятий, использует специальную терминологию дисциплины, не затрудняется при ответах на видоизмененные вопросы, сопровождает ответ примерами.

**Оценка «хорошо»** ставится, если:

- выполнены требования к оценке «отлично», но допущены 2 – 3 недочета при выполнении практических заданий и студент может их исправить самостоятельно или при небольшой помощи преподавателя;
- в письменном отчете по работе делает незначительные ошибки;

- при ответах на контрольные вопросы не допускает серьезных ошибок, легко устраняет отдельные неточности, но затрудняется в применении знаний в новой ситуации, приведении примеров.

**Оценка «удовлетворительно»** ставится, если:

- практическая работа выполнена не полностью, но объем выполненной части позволяет получить правильные результаты и выводы;
- в ходе выполнения работы студент продемонстрировал слабые практические навыки, были допущены ошибки;
- студент умеет применять полученные знания при решении простых задач по готовому алгоритму;
- в письменном отчете по работе допущены ошибки;
- при ответах на контрольные вопросы правильно понимает их сущность, но в ответе имеются отдельные пробелы и при самостоятельном воспроизведении материала требует дополнительных и уточняющих вопросов преподавателя.

**Оценка «неудовлетворительно»** ставится, если:

- практическая работа выполнена не полностью и объем выполненной работы не позволяет сделать правильных выводов, у студента имеются лишь отдельные представления об изученном материале, большая часть материала не усвоена;
- в письменном отчете по работе допущены грубые ошибки, либо он вообще отсутствует;
- на контрольные вопросы студент не может дать ответов, так как не овладел основными знаниями и умениями в соответствии с требованиями программы.