**МИНОБРНАУКИ РОССИИ**
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ О.Г. Локтионова
«_5_» _12_____ 2025 г.

# МАШИННОЕ ОБУЧЕНИЕ И НЕЙРОСЕТЕВЫЕ МОДЕЛИ

Методические указания к проведению лабораторных занятий
для студентов направления подготовки
09.04.01 Информатика и вычислительная техника

Курск 2025 г.

УДК 004.8
Составитель Е.Н. Иванова

Рецензент
Доцент кафедры программной инженерии,
кандидат технических наук
*Т.Н. Конаныхина*

**Машинное обучение и нейросетевые модели** : методические указания к проведению лабораторных занятий для студентов направления подготовки 09.04.01 Информатика и вычислительная техника / Юго-Зап. гос. ун-т; сост.: Е.Н. Иванова. – Курск, 2025. – 19 с. – Библиограф.: с. 19.

Содержатся цель каждого занятия, краткая теория. Приводятся примерные задания для выполнения и их реализация на лабораторных занятиях. Охватывается полный цикл работы: от первичной обработки сырых данных до программирования и обучения глубоких нейросетевых моделей.

Методические указания соответствуют требованиям программ, утвержденным учебно-методическим объединением по направлению Информатика и вычислительная техника.

Предназначены для студентов очной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл.печ.л. Уч.-изд.л. . Тираж 20 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

## Лабораторная работа №1

## Подготовка данных

### Цель работы

Освоить первичные методы обработки сырых данных: очистку, обработку пропусков и выбросов.

### Краткие теоретические сведения

Качество данных напрямую влияет на качество итоговой модели. Реальный мир предоставляет данные, которые часто содержат пропуски, шумы и противоречия. Основные этапы подготовки включают:

– обработка пропусков (missing values). Удаление строк с пропусками или заполнение их (средним, медианой, модой или более сложными методами интерполяции);

– обработка выбросов (outliers). Обнаружение аномальных значений с помощью статистических методов (правило трех сигм, межквартильный размах) и их корректировка или удаление;

– очистка данных. Удаление дубликатов, исправление опечаток в категориальных признаках.

### Задание

1. Загрузить набор данных о пассажирах Титаника. Заполнить пропуски в колонке Age средним значением в зависимости от пола и класса пассажира.

2. В датасете с ценами на недвижимость обнаружить и обработать выбросы в колонке price с помощью метода межквартильного размаха (IQR).

3. Для логов сервера удалить все строки с пропущенными значениями и дубликаты записей.

4. В датасете с отзывами клиентов найти и исправить типовые опечатки в названиях городов.

5. Используя библиотеку pandas, выполнить интерполяцию

временного ряда с пропущенными данными.

**Пример реализации**

```
import pandas as pd
import numpy as np

# Создаем образец данных с пропусками
data = {'Возраст': [25, 30, np.nan, 35, 40, 120], 'Зарплата':
[50000, 60000, 55000, np.nan, 65000, 70000]}
df = pd.DataFrame(data)

print("Исходные данные:\n", df)

# 1. Заполнение пропусков средним значением
df['Возраст'].fillna(df['Возраст'].median(), inplace=True)
df['Зарплата'].fillna(df['Зарплата'].mean(), inplace=True)

# 2. Обнаружение выбросов (например, по возрасту > 100)
outliers = df[df['Возраст'] > 100].index
df.drop(outliers, inplace=True)

print("\nДанные после обработки:\n", df)
```

**Вопросы:**

1. Какие существуют стратегии обработки пропущенных значений?
2. Чем отличается удаление выбросов от их замены?
3. Как определить, является ли значение аномальным?

## Лабораторная работа №2

## Проектирование признаков (Feature Engineering)

### Цель работы

Изучить методы создания новых признаков из существующих данных для улучшения качества моделей.

### Краткие теоретические сведения

Feature Engineering – это процесс преобразования сырых данных в признаки, которые лучше всего представляют основную проблему для моделей машинного обучения .

Кодирование категориальных признаков – преобразование текстовых категорий в числа (One-Hot Encoding, Label Encoding).

Дискретизация (бининг) – разделение непрерывного признака на несколько интервалов.

Создание взаимодействий (interaction features) – произведение, сумма или разность существующих признаков.

### Задание

Для датасета с датами создать признаки: день недели, месяц, является ли день выходным.

2. Закодировать категориальный признак "Цвет" с помощью One-Hot Encoding.

3. Выполнить дискретизацию возраста (бининг) по группам: ребенок, взрослый, пожилой.

4. Создать новый признак как отношение двух числовых колонок (например, площадь комнаты = общая площадь / количество комнат).

5. Использовать полиномиальные признаки для задачи регрессии.

**Пример реализации**

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

df = pd.DataFrame({'Пол': ['М', 'Ж', 'М', 'Ж'], 'Доход': [100, 150, 200, 120]})

# One-Hot Encoding
encoder = OneHotEncoder(sparse_output=False)
encoded_sex = encoder.fit_transform(df[['Пол']])
encoded_df = pd.DataFrame(encoded_sex, columns=encoder.get_feature_names_out(['Пол']))

# Создание нового признака
df['Доход_тыс'] = df['Доход'] / 1000

result = pd.concat([df, encoded_df], axis=1)
print(result)
```

**Вопросы**

1. В чем разница между One-Hot Encoding и Label Encoding? Когда какой лучше применять?
2. Что такое дискретизация данных и зачем она нужна?
3. Как новые признаки могут помочь в борьбе с недообучением модели?

## Лабораторная работа №3

## Нормализация данных

### Цель работы

Изучить влияние нормализации на скорость и качество обучения моделей

### Краткие теоретические сведения

Алгоритмы машинного обучения (особенно градиентный спуск и методы, основанные на расстояниях) чувствительны к масштабу признаков. Если один признак измеряется в тысячах, а другой – в долях единицы, первый будет доминировать.

Стандартизация (StandardScaler). Приводит данные к среднему = 0 и стандартному отклонению = 1. $z = (x - \mu) / \sigma$. Не гарантирует, что данные попадут в конкретный интервал.

Нормализация (MinMaxScaler). Масштабирует данные в заданный диапазон (обычно [0, 1]). $x\_norm = (x - min) / (max - min)$. Чувствительна к выбросам.

Важное правило: fit() вызывается только на обучающей выборке. Для преобразования тестовой выборки используется только transform(), чтобы не подглядывать в тестовые данные.

### Задание

Применить StandardScaler к числовым признакам датасета Iris.

2. Применить MinMaxScaler и сравнить результаты с данными из задания 1.

3. Обучить модель K-ближайших соседей на немасштабированных и масштабированных данных, сравнить точность.

4. Объяснить, почему масштабирование критически важно для метода главных компонент (PCA).

5. Нужно ли масштабировать бинарные признаки (0/1)? Ответ обосновать.

**Пример реализации**

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np

data = np.array([[1000, 0.1], [2000, 0.2], [3000, 0.3]])

# Стандартизация
scaler_std = StandardScaler()
standardized = scaler_std.fit_transform(data)

# Min-Max нормализация
scaler_mm = MinMaxScaler()
normalized = scaler_mm.fit_transform(data)

print("Стандартизованные:\n", standardized)
print("Нормализованные [0,1]:\n", normalized)
```

**Вопросы:**

1. В чем разница между нормализацией и стандартизацией?
2. Почему нормализация важна для нейронных сетей?
3. Надо ли масштабировать целевую переменную (y) в задачах регрессии?

## Лабораторная работа №4

## Визуализация данных

### Цель работы

Научиться строить графики для разведочного анализа данных (EDA) и презентации результатов.

### Краткие теоретические сведения

Визуализация помогает понять распределения данных, выявить выбросы и взаимосвязи между признаками. Инструменты визуализации:

– гистограммы. Для понимания распределения одного признака;

– ящики с усами (box plots). Для визуализации выбросов и квартилей;

– диаграммы рассеяния (scatter plots). Для поиска взаимосвязи между двумя признаками;

– матрица корреляций (heatmap). Для визуализации линейных связей.

### Задание

Построить гистограммы для всех числовых признаков датасета "Boston Housing".

2. Создать scatter plot с раскраской по целевому признаку.

3. Визуализировать попарные зависимости признаков (pairplot).

4. Построить ящик с усами для обнаружения выбросов.

5. Визуализировать временной ряд (график цены акции от времени).

### Пример реализации

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

```
import seaborn as sns
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Гистограмма
plt.hist(df['sepal length (cm)'], bins=20)
plt.title('Распределение длины чашелистика')
plt.show()

# Матрица корреляций
sns.heatmap(df.corr(), annot=True)
plt.show()
```

**Вопросы:**

1. Какой график лучше всего подходит для отображения корреляции?
2. Как по гистограмме понять, что данные имеют асимметрию (скошенность)?
3. Что такое "многомерная визуализация"?

## Лабораторная работа №5

## Алгоритмы обучения с учителем

### Цель работы

Реализовать и обучить модели классификации и регрессии.

### Краткие теоретические сведения

Обучение с учителем (Supervised Learning) – это подход, при котором модель обучается на размеченных данных, где для каждого примера известен правильный ответ.

Регрессия. Предсказание непрерывного значения (например, цены дома). Алгоритмы: линейная регрессия, решающие деревья.

Классификация. Предсказание категории (например, спам/не спам). Алгоритмы: Логистическая регрессия, метод опорных векторов (SVM), K-ближайших соседей (KNN).

Оценка качества. Accuracy, Precision, Recall, F1-мера для классификации; MSE, MAE, $R^2$ для регрессии.

### Задание

1. Обучить логистическую регрессию для бинарной классификации.
2. Обучить линейную регрессию для предсказания стоимости автомобиля .
3. Сравнить качество работы дерева решений и случайного леса на датасете "Wine".
4. Подобрать гиперпараметры для KNN с помощью GridSearchCV.
5. Решить задачу многоклассовой классификации.

### Пример реализации

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score

# X, y - признаки и метки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

model = RandomForestClassifier()
model.fit(X_train, y_train) # Обучение

predictions = model.predict(X_test)
print(f"Точность: {accuracy_score(y_test, predictions)}")
```

**Вопросы:**

1. Чем отличается переобучение (overfitting) от недообучения (underfitting)?
2. Для чего нужна кросс-валидация?
3. В каких случаях лучше использовать метрику точности (Accuracy), а в каких – полноту (Recall)?

## Лабораторное занятие №6

## Алгоритмы обучения без учителя

### Цель работы

Применить методы кластеризации и снижения размерности для поиска структуры в данных.

### Краткие теоретические сведения

Обучение без учителя (Unsupervised Learning) работает с неразмеченными данными. Модель ищет скрытые закономерности самостоятельно.

Кластеризация. Группировка объектов по схожим характеристикам. Алгоритмы: K-means, иерархическая кластеризация.

Снижение размерности. Уменьшение количества признаков с минимальной потерей информации. Алгоритмы: PCA (метод главных компонент)

### Задание

Выполнить кластеризацию клиентов магазина по данным покупок.

2. Использовать метод локтя (elbow method) для определения оптимального числа кластеров.

3. Сократить размерность датасета цифр MNIST с 784 признаков до 2 с помощью PCA и визуализировать результат.

4. Сравнить результаты кластеризации K-means и DBSCAN.

5. Реализовать сжатие изображения с помощью K-means (уменьшение количества цветов).

### Пример реализации

```
from sklearn.cluster import KMeans
import numpy as np
```

```
X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])

kmeans = KMeans(n_clusters=2, random_state=0, n_init="auto")
kmeans.fit(X)
print(f"Метки кластеров: {kmeans.labels_}")
print(f"Центры кластеров: {kmeans.cluster_centers_}")
```

**Вопросы:**

1. Чем кластеризация отличается от классификации?
2. Как интерпретировать результат работы PCA?
3. Какой алгоритм кластеризации лучше подходит для данных произвольной формы?

**Лабораторное занятие №7**

**Программирование нейронной сети с контролируемым обучением**

**Цель работы**

Реализовать полносвязную нейронную сеть (многослойный перцептрон) для задачи классификации.

**Краткие теоретические сведения**

Искусственный нейрон – это математическая модель, имитирующая работу биологического нейрона. Он принимает входные сигналы, умножает их на веса, суммирует и пропускает через функцию активации.

Архитектура: входной слой, скрытые слои, выходной слой.

Функции активации: ReLU, Sigmoid, Tanh.

Алгоритмы обучения: обратное распространение ошибки (Backpropagation) и градиентный спуск .

**Задание**

Реализовать перцептрон для логической операции "И".

2. Написать класс многослойного перцептрона с нуля (без высокоуровневых библиотек).

3. Обучить нейросеть на датасете Iris.

4. Исследовать влияние скорости обучения (learning rate) на сходимость.

5. Добавить L2-регуляризацию для борьбы с переобучением.

**Пример реализации**

```
import torch
import torch.nn as nn
import torch.optim as optim
```

```
class SimpleNN(nn.Module):
   def init(self):
      super().init()
      self.fc1 = nn.Linear(784, 128)  # Вход 784 -> скрытый 128
      self.relu = nn.ReLU()
      self.fc2 = nn.Linear(128, 10)   # Выход 10 классов
def forward(self, x):
      x = self.fc1(x)
      x = self.relu(x)
      x = self.fc2(x)
      return x

model = SimpleNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# Далее следует цикл обучения (forward, loss, backward, step)
```

**Вопросы:**

1. Объясните суть метода обратного распространения ошибки.
2. Зачем нужны нелинейные функции активации?
3. Что происходит, если обнулить все веса в сети?

## Лабораторное занятие №8

## Обучение глубокой сети

### Цель работы

Познакомиться с современными подходами к обучению глубоких нейронных сетей и проблемой исчезающего градиента.

### Краткие теоретические сведения

Глубокое обучение (Deep Learning) – это подмножество машинного обучения, использующее нейронные сети с множеством слоев.

Проблемы глубоких сетей: исчезающий/взрывающийся градиент, переобучение.

Сверточные сети (CNN). Эффективны для обработки изображений.

Регуляризация. Dropout, Batch Normalization.

Современные архитектуры: остаточные сети (ResNet) позволяют обучать очень глубокие модели.

### Задание

Обучить сверточную нейросеть (CNN) для классификации рукописных цифр MNIST.

2. Использовать предобученную модель (transfer learning) для классификации котиков и собачек.

3. Сравнить глубину сети (2 слоя против 10 слоев) на качество обучения.

4. Применить Batch Normalization и посмотреть, как это влияет на скорость сходимости.

5. Визуализировать карты активации сверточной сети.

### Пример реализации

class DeepNN(nn.Module):

```
    def init(self):
        super().init()
        self.layers = nn.Sequential(
            nn.Linear(784, 1024),
            nn.ReLU(),
            nn.Dropout(0.5),      # 50% нейронов выключается случайно
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        return self.layers(x)
```

**Вопросы:**

1. Что такое проблема исчезающего градиента?
2. Зачем в глубоких сетях нужны "skip connections" (обходные соединения), как в ResNet?
3. Чем CNN отличается от полносвязной сети при работе с изображениями?

**Список использованных источников**

1. Учебник по машинному обучению : онлайн-учебник // Школа анализа данных : [сайт] / АНО ДПО «Образовательные технологии Яндекса». – [2007-2023]. – URL: https://education.yandex.ru/handbook/ml (дата обращения: 20.12.2025).

2. Гарбук, С. В. Функциональность и безопасность систем искусственного интеллекта: качество данных / С. В. Гарбук. // Открытые системы. СУБД. – 2024. – № 1. – С. 15-20. – Текст : непосредственный

3. Бринк, Х. Машинное обучение / Х. Бринк, Д. Ричардс, М. Феверолф. — Санкт-Петербург : Питер, 2017. – 336 с. – ISBN 978-5-496-02570-3. – Текст : непосредственный.

4. ГОСТ Р 71484.1-2024. Искусственный интеллект. Качество данных для аналитики и машинного обучения. Часть 1. Обзор, терминология и примеры. – Москва : Стандартинформ, 2024. – 24 с.

5. ГОСТ Р 71484.4-2024. Искусственный интеллект. Качество данных для аналитики и машинного обучения. Часть 4. Структура процесса управления качеством данных. – Москва : Стандартинформ, 2024. – 16 с. ·

6. ГОСТ Р 59276-2020. Системы искусственного интеллекта. Способы обеспечения доверия. Общие положения. – Москва : Стандартинформ, 2021. – 16 с.