Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №3**

По дисциплине: «Анализ больших данных»

Тема: «**Бинарная и многоклассовая классификация.**

**Оценка качества задачи классификации**»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС - 21.06  Леонов А. М. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2025

**Оглавление**

[**Введение** 3](#_Toc191336105)

[**Цель:** 3](#_Toc191336106)

[**Краткий опрос по теоретической части:** 3](#_Toc191336107)

[**Формулировка задач:** 3](#_Toc191336108)

[**Задача 1. Разбор данных из Kaggle** 3](#_Toc191336109)

[**Задача 2. Задание по варианту** 4](#_Toc191336110)

[**Ссылка на репозиторий с программной реализацией:** 5](#_Toc191336111)

[**Описание проделанной работы** 6](#_Toc191336112)

[**Ответы на вопросы:** 6](#_Toc191336113)

[**Описание алгоритмов:** 9](#_Toc191336114)

[**Результаты** 12](#_Toc191336115)

[**Общее задание** 12](#_Toc191336116)

[**Описание базы данных – признаки, целевая переменная:** 12](#_Toc191336117)

[**Исследовательский анализ данных:** 12](#_Toc191336118)

[**Вывод по анализу:** 13](#_Toc191336119)

[**Визуализация результата:** 14](#_Toc191336120)

[**Критерии качества:** 15](#_Toc191336121)

[**Вывод по результату и критериям:** 15](#_Toc191336122)

[**Самостоятельное задание по варианту** 16](#_Toc191336123)

[**Описание базы данных – признаки, целевая переменная:** 16](#_Toc191336124)

[**Исследовательский анализ данных:** 17](#_Toc191336125)

[**Вывод по анализу:** 18](#_Toc191336126)

[**Визуализация результата:** 19](#_Toc191336127)

[**Критерии качества:** 20](#_Toc191336128)

[**Вывод по результату и критериям:** 20](#_Toc191336129)

[**Вывод** 21](#_Toc191336130)

[**Использованные материалы** 22](#_Toc191336131)

[**Листинг кода** 23](#_Toc191336132)

# **Введение**

## **Цель:**

Изучение на практике методов data mining для решения задач классификации и оценке качества классификации.

## **Краткий опрос по теоретической части:**

В данном разделе необходимо ответить на вопросы.

1. Что такое классификация?
2. Чем отличается бинарная классификация от многоклассовой?
3. Какие преимущества имеет логистическая регрессия?
4. Какую задачу решает логистическая регрессия?
5. Как исключить входные факторы (признаки) с низкой значимостью?
6. Как оценить точность классификации?
7. Как работает алгоритм KNN? Чем отличается алгоритм от логистической регрессии?
8. Что показывают критерии качества и ROC-кривая для задачи с определением пола?
9. Какие входные и выходные параметры используются в модели Логистической регрессии?
10. Что показывают критерии качества и ROC-кривая для задачи с определением доброкачественной опухоли?
11. Как построить таблицу сопряженности (confusion matrix)?

## **Формулировка задач:**

## **Задача 1. Разбор данных из Kaggle**

Построить минимум две модели классификации с использованием алгоритмов KNN, логистическая регрессия, SVM. Алгоритмы допустимо использовать из библиотеки scikit-learn.

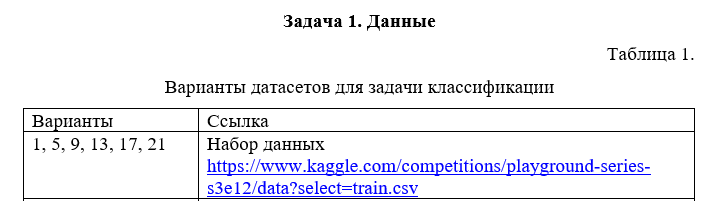
Модели строятся для задачи бинарной классификации.

Затем сравните их качество, используя метрики, такие как A, P, R, E, confusion\_matrix, ROC\_AUC.

на тестовых данных.

Подробное описание:

1. Загрузить один из наборов данных. Загружаются все файлы по ссылке.
2. Провести разведочный анализ данных, ответив на следующие вопросы:
3. Сколько строк в датафрейме, сколько столбцов
4. Сколько места занимает датафрейм в оперативной памяти
5. Для каждой интервальной переменной подсчитать следующее - мин, медиана, среднее, макс и персентили 25, 75
6. Для каждой категориальной переменной рассчитать моду и сколько раз мода встречается в данных
7. Подготовка датасета к построению моделей ML
8. Провести анализ и обработку пропусков (либо заменить, либо удалить)
9. Провести анализ и обработку выбросов (либо заменить, либо удалить)
10. Провести анализ и обработку категориальных переменных (сколько таких переменных, закодируйте категориальные переменные одним из методов ( one hot encoding, mean target, frequence encoding)
11. Разделить датасет на трейн и тест
12. Построить следующие классификационные алгоритмы:
13. Knn
14. Logistic regression
15. SVM
16. Оценить качество алгоритмов, выбрать самый оптимальный алгоритм
17. Возможно ли улучшить алгоритм, предложите идеи?



## **Задача 2. Задание по варианту**

**Вариант и условия задачи:**

**Вариант 9**

БД- Sample databases for the SQL Server

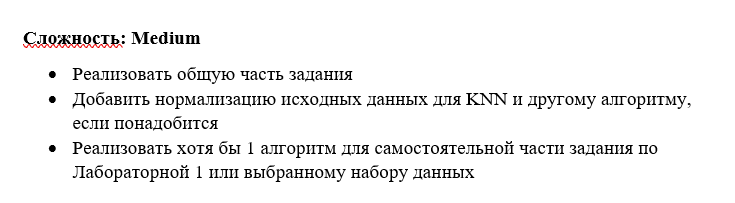
Ссылка на БД- <https://www.kaggle.com/datasets/emrahaydemr/sample-databases-for-the-sql-server-course>

В качестве индивидуального набора данных взять данных из Лабораторной работы 1. Выполнить те же действия, что в Задаче 1.

Если по какой-то причине данные взять для анализа нельзя, то выбрать набора данных из представленных ниже. Для данных ниже описать какие есть столбцы, какие из них признаки и какой целевая переменная перед исследованием.

В качестве датасета была выбрана база из 1 лабораторной работы.

Лабораторная работы выполнена на сложность Medium



## **Ссылка на репозиторий с программной реализацией:**

GitHub- https://github.com/akkafe1ix/BigData\_Lab\_3

# **Описание проделанной работы**

## **Ответы на вопросы:**

1. **Что такое классификация?** Классификация — это задача машинного обучения, в которой необходимо отнести объект к одной из заранее определённых категорий или классов, основываясь на его признаках. Например, классификация писем как "спам" или "не спам".
2. **Чем отличается бинарная классификация от многоклассовой?**

* Бинарная классификация решает задачу разделения объектов на два класса (например, "да" и "нет", "0" и "1").
* Многоклассовая классификация подразумевает разделение объектов на более чем два класса (например, классификация животных по видам: кошка, собака, птица).

1. **Какие преимущества имеет логистическая регрессия?**

* Простота и интерпретируемость модели.
* Меньше вычислительных затрат по сравнению с другими моделями.
* Хорошо работает при линейных зависимостях между признаками и целевой переменной.
* Легко расширяется для многоклассовой классификации (с использованием методов One-vs-Rest или Multinomial Logistic Regression).

1. **Какую задачу решает логистическая регрессия?** Логистическая регрессия решает задачу классификации, предсказывая вероятность того, что объект принадлежит к определённому классу. Обычно используется для бинарной классификации, но может быть адаптирована и для многоклассовых задач.
2. **Как исключить входные факторы (признаки) с низкой значимостью?** Для исключения признаков с низкой значимостью можно:

* Использовать методы отбора признаков, такие как рекурсивное исключение признаков (RFE) или регуляризацию (например, L1-регуляризация).
* Применить методы на основе важности признаков, такие как деревья решений или случайные леса.
* Проводить анализ корреляции и исключать сильно коррелированные признаки.

1. **Как оценить точность классификации?** Точность классификации можно оценить с помощью:

* Матрицы ошибок (confusion matrix) — показывает количество верно и неверно классифицированных объектов по каждому классу.
* Точности (accuracy) — доля правильно классифицированных объектов от общего числа объектов.
* Коэффициента F1, точности (precision) и полноты (recall).
* ROC-кривой и AUC.

1. **Как работает алгоритм KNN? Чем отличается алгоритм от логистической регрессии?** Алгоритм K-ближайших соседей (KNN) классифицирует объект на основе класса ближайших к нему объектов в пространстве признаков. Чем ближе объекты друг к другу, тем более вероятно, что они принадлежат к одному классу. Алгоритм является непараметрическим и неопределённым, что означает, что он не строит явную модель, а делает предсказания на основе соседей.

В отличие от логистической регрессии, которая предполагает линейную зависимость между признаками и результатом, KNN основывается на расстояниях между объектами.

1. **Что показывают критерии качества и ROC-кривая для задачи с определением пола?** Для задачи классификации по полу ROC-кривая показывает способность модели различать два класса (например, "мужчина" и "женщина"). AUC (площадь под ROC-кривой) показывает, насколько хорошо модель может отличать один класс от другого. Чем выше AUC, тем лучше модель. Критерии качества (такие как точность, полнота, F1-меры) показывают, насколько модель верно классифицирует объекты каждого класса.
2. **Какие входные и выходные параметры используются в модели Логистической регрессии?**

* Входные параметры: Признаки объекта (например, возраст, пол, уровень образования и т.д.).
* Выходной параметр: Вероятность принадлежности объекта к определённому классу. Обычно для бинарной классификации выходной параметр — это вероятность того, что объект принадлежит к положительному классу.

1. **Что показывают критерии качества и ROC-кривая для задачи с определением доброкачественной опухоли?** Критерии качества и ROC-кривая для задачи с определением доброкачественной опухоли помогут оценить способность модели правильно классифицировать опухоли как доброкачественные или злокачественные. ROC-кривая будет показывать компромисс между полнотой и ложными срабатываниями для различных пороговых значений. AUC будет показывать общую способность модели различать классы опухолей.
2. **Как построить таблицу сопряженности (confusion matrix)?** Таблица сопряженности (confusion matrix) — это таблица, которая позволяет оценить точность классификации, показывая количество верных и неверных предсказаний для каждого класса. Она имеет вид:

Predicted Negative | Predicted Positive

Actual Negative TN (True Negative) | FP (False Positive)

Actual Positive FN (False Negative) | TP (True Positive)

Можно построить её с помощью функции confusion\_matrix из библиотеки sklearn:

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_true, y\_pred)

print(cm)

## **Описание алгоритмов:**

**1. K-Nearest Neighbors (KNN)**

**KNN** — это алгоритм классификации (или регрессии), который основывается на принципе, что объекты, расположенные близко друг к другу, имеют схожие характеристики. Это один из самых простых алгоритмов для классификации и регрессии, но с высокими требованиями к вычислительным ресурсам на больших данных.

**Основные принципы работы:**

* Алгоритм работает следующим образом: для каждого нового объекта алгоритм находит **K ближайших соседей** в обучающем наборе данных и на основе их меток классифицирует этот объект.
* **Метрика расстояния**: Расстояние между объектами чаще всего измеряется с использованием евклидова расстояния, хотя могут быть использованы и другие метрики (например, манхэттенское расстояние, косинусное сходство).

**Шаги алгоритма:**

1. Выбирается значение **K** (количество ближайших соседей).
2. Для каждого нового примера (классифицируемого объекта) вычисляется расстояние до всех объектов обучающего набора.
3. Выбираются **K** объектов с наименьшими расстояниями.
4. Для классификации выбирается наиболее частая метка среди **K** ближайших соседей (для регрессии — среднее значение).

**2. Logistic Regression**

**Логистическая регрессия** — это алгоритм машинного обучения, используемый для **бинарной классификации**. Несмотря на название, это не регрессия в традиционном смысле, а именно классификационный метод, который использует функцию логистической регрессии для предсказания вероятности того, что объект принадлежит к одному из двух классов.

**Основные принципы работы:**

* Логистическая регрессия вычисляет вероятность принадлежности объекта к положительному классу (например, 1 в задаче бинарной классификации) с помощью логистической функции (сигмоиды).
* Функция сигмоиды преобразует линейную комбинацию признаков в диапазон от 0 до 1, что и интерпретируется как вероятность:

Изображение выглядит как текст, Шрифт, линия, чек

Контент, сгенерированный ИИ, может содержать ошибки.

* Алгоритм минимизирует **логистическую ошибку** с использованием градиентного спуска для оптимизации коэффициентов www.

**Шаги алгоритма:**

1. Модель логистической регрессии находит оптимальные коэффициенты www, которые минимизируют ошибку в предсказаниях.
2. Эти коэффициенты затем используются для вычисления вероятности того, что объект принадлежит к положительному классу.
3. Если вероятность больше 0.5, объект классифицируется как положительный (1), иначе как отрицательный (0).

**3. Support Vector Machine (SVM)**

**SVM** — это алгоритм машинного обучения, используемый для классификации и регрессии. Он находит оптимальную гиперплоскость, которая максимально разделяет объекты разных классов в пространстве признаков.

**Основные принципы работы:**

* Алгоритм ищет гиперплоскость, которая создает максимальный зазор (margin) между объектами разных классов.
* Объекты, которые находятся ближе всего к гиперплоскости, называются опорными векторами, и именно они определяют положение разделяющей границы.
* В случае, если данные неразделимы линейно, используется ядровая функция, которая переводит данные в пространство большей размерности, где они становятся линейно разделимыми.

**Шаги алгоритма:**

1. Определяется оптимальная гиперплоскость, которая разделяет классы с максимальным зазором.
2. Если данные не линейно разделимы, применяется ядровая функция (например, полиномиальное ядро или ядро Гаусса).
3. Новые объекты классифицируются на основе их положения относительно гиперплоскости.

SVM особенно эффективен на небольших наборах данных с высокой размерностью, но может требовать высокой вычислительной мощности при использовании сложных ядерных функций.

# **Результаты**

## **Общее задание**

### **Описание базы данных – признаки, целевая переменная:**

Набор данных **Playground Series - Season 3, Episode 12** с платформы Kaggle содержит информацию о физических и химических свойствах образцов, используемых для бинарной классификации.

**Признаки (Features):**

Датасет включает 7 числовых признаков, описывающих различные характеристики образцов:

1. gravity – плотность образца.
2. ph – уровень pH.
3. osmo – осмолярность (концентрация растворённых частиц).
4. cond – электрическая проводимость.
5. urea – концентрация мочевины.
6. calc – уровень кальция.
7. id – уникальный идентификатор (не используется в моделировании).

Все признаки являются числовыми и измеряются в соответствующих единицах.

**Целевая переменная (Target):**

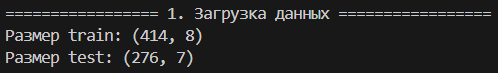
Целевая переменная **target** представляет собой бинарную классификацию:

* 0: Отрицательный результат.
* 1: Положительный результат.

Задача классификации заключается в предсказании значения target на основе представленных признаков.

### **Исследовательский анализ данных:**

A. Сколько строк в датафрейме, сколько столбцов

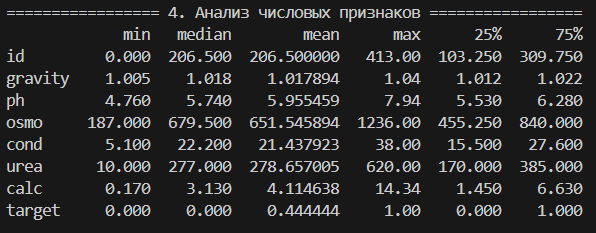


B. Сколько места занимает датафрейм в оперативной памяти



C. Для каждой интервальной переменной подсчитать следующее - мин, медиана, среднее, макс и персентили 25, 75





D. Для каждой категориальной переменной рассчитать моду и сколько раз мода встречается в данных



### **Вывод по анализу:**

**Размерность датафрейма:**

* В датафрейме **414 строк и 8 столбцов**.
* Семь столбцов содержат числовые признаки, а один столбец (**target**) является **целевой переменной** (0 или 1).

**Место, занимаемое в памяти:**

* Датафрейм занимает **около 26 KB** памяти.
* Малый объем данных позволяет **эффективно анализировать и обрабатывать** их без значительных вычислительных затрат.

**Статистики для числовых признаков:**

* **Все числовые признаки находятся в разумных пределах.**
* **Средние и медианные значения** показывают, что данные распределены **без значительных выбросов**.

**Категориальные признаки:**

* В датасете **отсутствуют категориальные признаки** – все данные числовые.

**Пропущенные значения:**

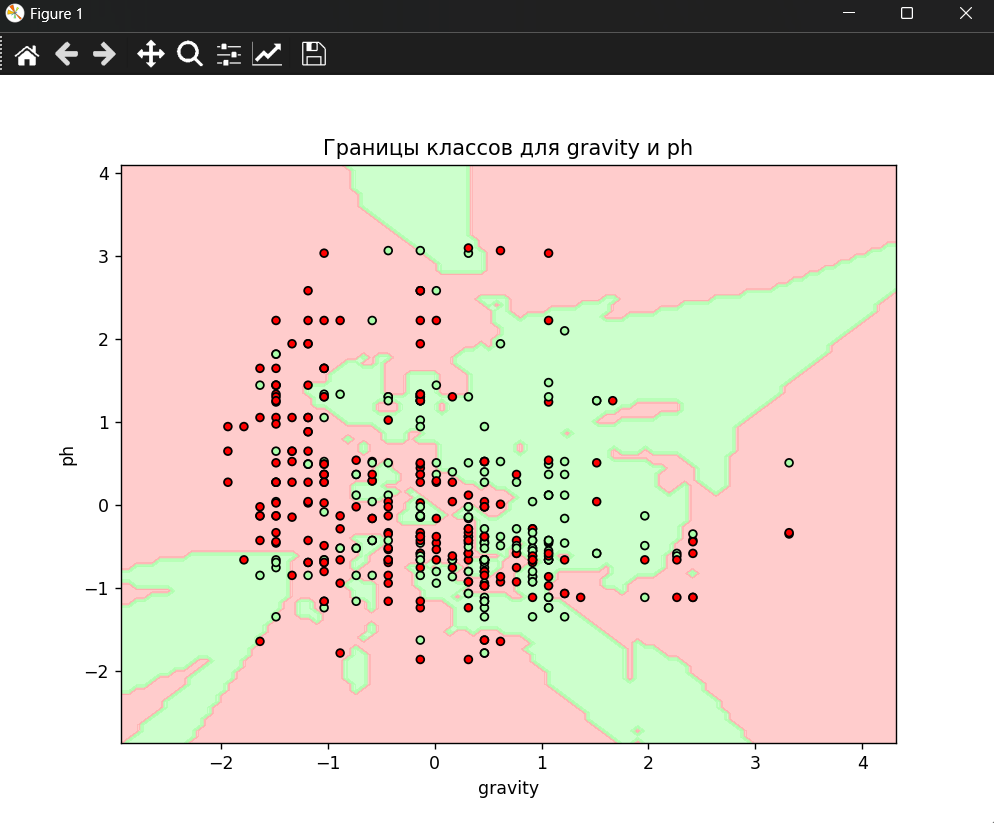
* **Пропущенные значения отсутствуют**, данные полностью заполнены.

**Мода для целевой переменной:**

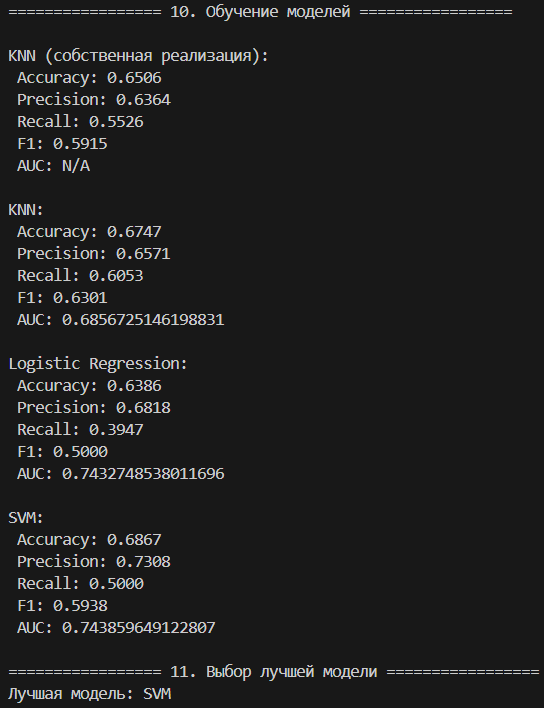
* Для целевой переменной **target** мода — **0**, что означает, что **класс 0 встречается чаще всего**.

### **Визуализация результата:**

**KNN**



### **Критерии качества:**



### **Вывод по результату и критериям:**

SVM оказалась более точной, чем KNN из библиотеки, КНН собственной реализации и чем логическая регрессия.

## **Самостоятельное задание по варианту**

### **Описание базы данных – признаки, целевая переменная:**

Набор данных содержит информацию о пользователях, включая их персональные данные, академические показатели и финансовые сведения.

**Признаки (Features):**

Датасет включает 21 признак, среди которых есть числовые и категориальные переменные:

1. id – уникальный идентификатор пользователя (не используется в моделировании).
2. adi – имя пользователя.
3. soyadi – фамилия пользователя.
4. eposta – электронная почта.
5. ceptel – номер мобильного телефона.
6. dogumtarihi – дата рождения.
7. dogumyeri\_tilcelerid – код района рождения.
8. cinsiyet – пол (булева переменная).
9. tckimlikno – идентификационный номер.
10. kangrubu\_tkangruplariid – код группы крови.
11. группа крови – группа крови (категориальный признак).
12. район рождения – название района рождения.
13. город рождения – название города рождения.
14. факультет – название факультета (пропуски: 1584).
15. отделение – название учебного отделения (пропуски: 1584).
16. курс – название курса (пропуски: 3544).
17. оценка за промежуточный экзамен – числовой признак (разброс: 1–100, пропуски: 3544).
18. оценка за финальный экзамен – числовой признак (разброс: 1–100, пропуски: 3544).
19. оценка (буквенная) – буквенная оценка (все значения отсутствуют).
20. оплата за летнюю школу – сумма оплаты (разброс: 105–1244, пропуски: 4141).
21. пол – текстовое обозначение пола (например, «мужчина»).

Все числовые признаки измеряются в соответствующих единицах.

**Целевая переменная (Target):**

**пол** – бинарная классификация:

* **мужчина**
* **женщина**

Задача классификации заключается в предсказании значения пола на основе представленных признаков.

### **Исследовательский анализ данных:**

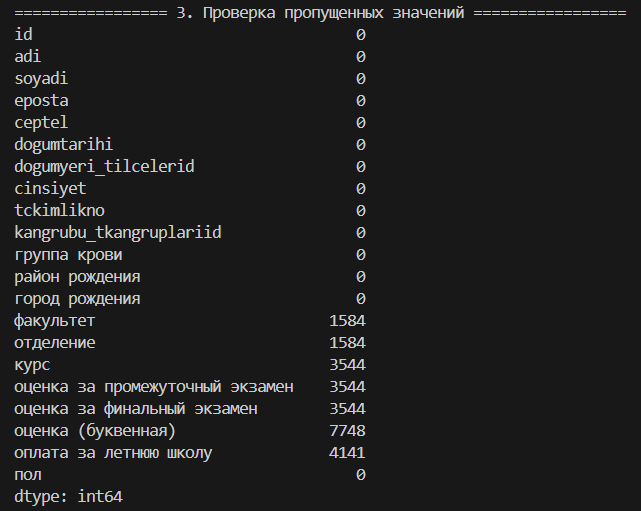
A. Сколько строк в датафрейме, сколько столбцов

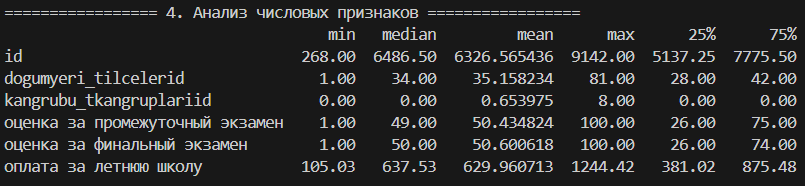


B. Сколько места занимает датафрейм в оперативной памяти

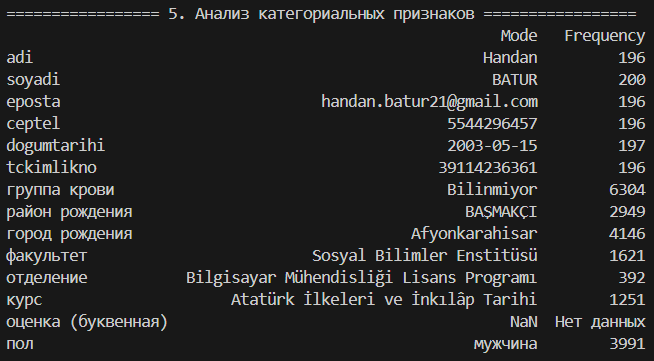


C. Для каждой интервальной переменной подсчитать следующее - мин, медиана, среднее, макс и персентили 25, 75





D. Для каждой категориальной переменной рассчитать моду и сколько раз мода встречается в данных



### **Вывод по анализу:**

**Размерность датафрейма:**

* В датафрейме 7748 строк и 21 столбец.
* Данные включают числовые и категориальные признаки.
* Целевая переменная (пол) является категориальной и принимает два значения: "мужчина" и "женщина".

**Место, занимаемое в памяти:**

* Датафрейм занимает около 1.2 MB памяти.
* Несмотря на относительно большой объем данных, их можно эффективно анализировать и обрабатывать.

**Статистики для числовых признаков:**

* Числовые признаки, такие как оценки за экзамены и стоимость летней школы, находятся в разумных пределах.
* Средние и медианные значения показывают, что данные распределены относительно равномерно, без значительных выбросов.

**Категориальные признаки:**

* Датасет содержит несколько категориальных признаков, включая факультет, отделение, курс, группу крови, место рождения и саму целевую переменную.
* Наиболее частая категория для факультета – Sosyal Bilimler Enstitüsü, для города рождения – Afyonkarahisar.

**Пропущенные значения:**

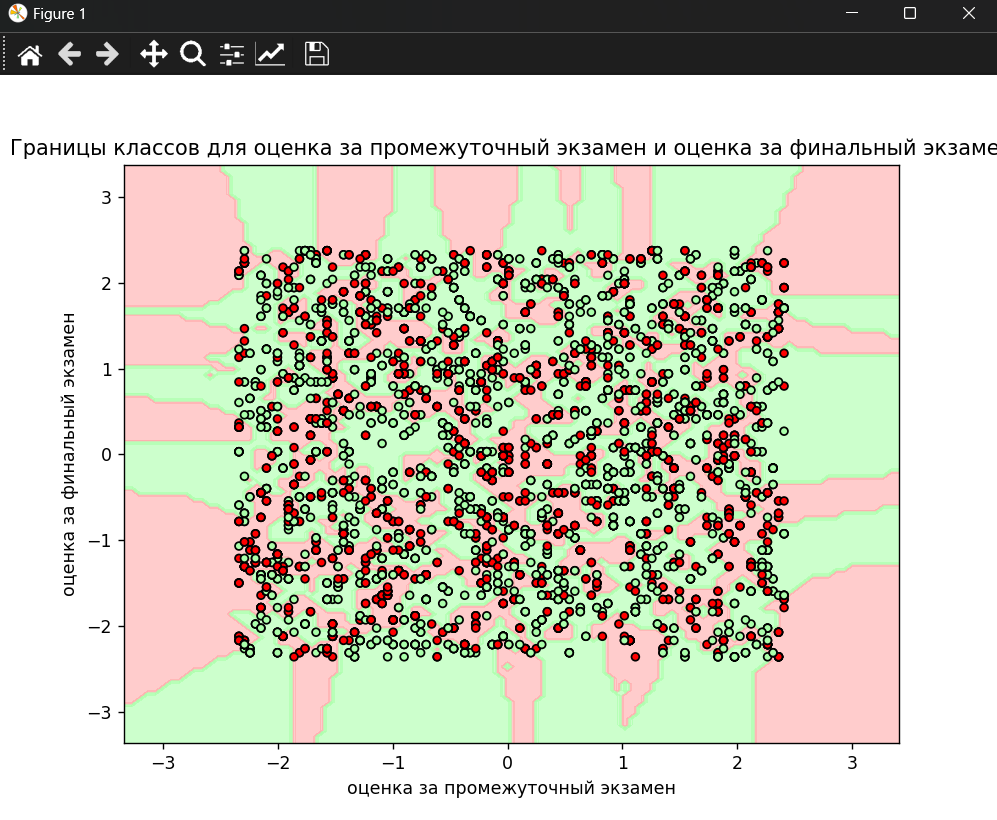
* Пропущенные значения присутствуют в следующих столбцах:
* факультет и отделение – 1584 пропущенных значения.
* курс, оценка за экзамены – 3544 пропущенных значения.
* оплата за летнюю школу – 4141 пропущенное значение.
* оценка (буквенная) отсутствует полностью.
* Остальные столбцы полностью заполнены.

**Мода для целевой переменной:**

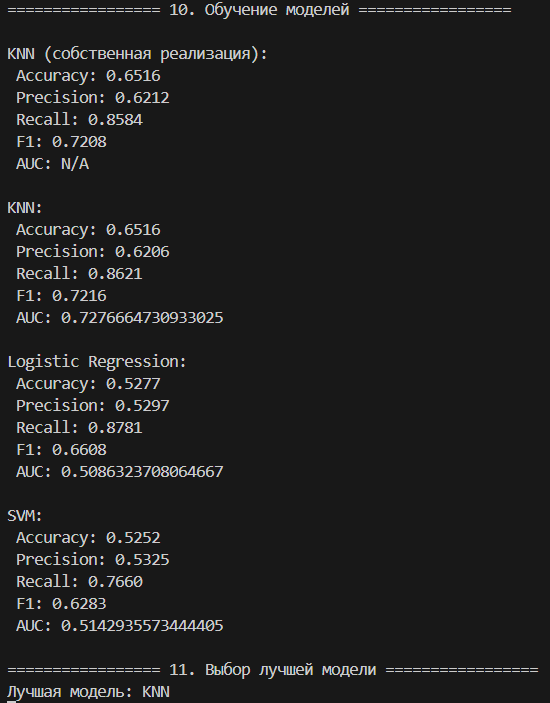
* Наиболее часто встречающееся значение (мода) для целевой переменной ("пол") – "мужчина". Это означает, что в выборке больше мужчин, чем женщин.

### **Визуализация результата:**

**KNN**



### **Критерии качества:**



### **Вывод по результату и критериям:**

Библиотечный KNN оказался более точным, чем SVM, КНН собственной реализации и чем логическая регрессия.

# **Вывод**

Лабораторная работа позволила на практике освоить основные методы классификации и оценки качества моделей. Анализ данных, предобработка, выбор алгоритмов и их настройка показали значительное влияние подготовки данных на итоговую точность классификации. Оптимальный алгоритм определяется не только метриками качества, но и особенностями конкретного датасета.

Программа успешно загружает и анализирует данные, вычисляя основные статистические характеристики, выявляя пропуски и выбросы, а также преобразуя категориальные признаки. Были построены и сравнены несколько моделей классификации, включая KNN, логистическую регрессию и SVM. Анализ показал, что наибольшую точность продемонстрировал алгоритм SVM, однако выбор оптимальной модели зависит от особенностей данных и метрик качества.

# **Использованные материалы**

1. Введение в библиотеку Pandas // PythonRu URL: https://pythonru.com/biblioteki/pandas-vvedenie (дата обращения: 24.02.2025).
2. Основы визуализации данных в Matplotlib и Seaborn // Habr URL: https://habr.com/ru/post/485370 (дата обращения: 24.02.2025).
3. Полное руководство по SQLAlchemy // PythonRu URL: <https://pythonru.com/biblioteki/vvedenie-v-sqlalchemy> (дата обращения: 24.02.2025).
4. Статистический анализ данных в Python с использованием SciPy // DataScienceBlog URL: https://datascienceblog.net/post/statistics/scipy (дата обращения: 24.02.2025).
5. Реализация градиентного спуска в Python // Towards Data Science URL: https://towardsdatascience.com/gradient-descent (дата обращения: 24.02.2025).
6. Разведочный анализ данных (EDA) в Python // AnalyticsVidhya URL: https://www.analyticsvidhya.com/blog/2020/02/exploratory-data-analysis-in-python (дата обращения: 24.02.2025).
7. Работа с датасетом mpg в Seaborn // Seaborn Documentation URL: https://seaborn.pydata.org/generated/seaborn.load\_dataset.html (дата обращения: 24.02.2025).
8. Метод опорных векторов (SVM) // GeeksforGeeks URL: https://www.geeksforgeeks.org/support-vector-machine-introduction/ (дата обращения: 24.02.2025).
9. Логистическая регрессия в машинном обучении // AnalyticsVidhya URL: https://www.analyticsvidhya.com/blog/2021/06/logistic-regression/ (дата обращения: 24.02.2025).
10. Основы KNN-алгоритма // Towards Data Science URL: https://towardsdatascience.com/k-nearest-neighbors (дата обращения: 24.02.2025).

# **Листинг кода**

Часть 1.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

from matplotlib.colors import ListedColormap

from collections import Counter

#  Реализация KNN с нуля

class KNN:

    def \_\_init\_\_(self, k=5):

        """ Инициализация модели с параметром k (количество ближайших соседей). """

        self.k = k

    def fit(self, X\_train, y\_train):

        """ Запоминаем обучающие данные. """

        self.X\_train = np.array(X\_train)

        self.y\_train = np.array(y\_train)

    def predict(self, X\_test):

        """ Для каждого объекта из тестового набора находим предсказанный класс. """

        predictions = [self.\_predict(x) for x in np.array(X\_test)]

        return np.array(predictions)

    def \_predict(self, x):

        """ Находим класс для одного объекта x. """

        distances = [self.\_euclidean\_distance(x, x\_train) for x\_train in self.X\_train]

        k\_indices = np.argsort(distances)[:self.k]

        k\_nearest\_labels = [self.y\_train[i] for i in k\_indices]

        most\_common = Counter(k\_nearest\_labels).most\_common(1)[0][0]

        return most\_common

    def \_euclidean\_distance(self, x1, x2):

        """ Вычисление евклидова расстояния между двумя точками. """

        return np.sqrt(np.sum((x1 - x2) \*\* 2))

#  Загрузка данных

print("\n================= 1. Загрузка данных =================")

try:

    train = pd.read\_csv("C:\\Users\\Алексей\\Desktop\\8 Семестр\\(Корнеева) Анализ больших данных\\Lab\\BigData\_Lab\_3\\data\\train.csv")

    test = pd.read\_csv("C:\\Users\\Алексей\\Desktop\\8 Семестр\\(Корнеева) Анализ больших данных\\Lab\\BigData\_Lab\_3\\data\\test.csv")

    print(f"Размер train: {train.shape}")

    print(f"Размер test: {test.shape}")

except Exception as e:

    print(f"Ошибка загрузки данных: {e}")

    exit()

print("\n================= 2. Информация о данных =================")

print(train.info() if not train.empty else "Данные train отсутствуют.")

print("\n================= 3. Проверка пропущенных значений =================")

if not train.empty:

    missing\_values = train.isnull().sum()

    print(missing\_values if missing\_values.sum() > 0 else "Пропущенные значения отсутствуют.")

else:

    print("Данные train отсутствуют.")

print("\n================= 4. Анализ числовых признаков =================")

numerical\_features = train.select\_dtypes(include=['float64', 'int64']).columns.tolist()

if numerical\_features:

    numerical\_stats = train[numerical\_features].agg(['min', 'median', 'mean', 'max']).T

    numerical\_stats["25%"] = train[numerical\_features].apply(lambda x: np.percentile(x.dropna(), 25))

    numerical\_stats["75%"] = train[numerical\_features].apply(lambda x: np.percentile(x.dropna(), 75))

    print(numerical\_stats)

else:

    print("Числовые признаки отсутствуют.")

print("\n================= 5. Анализ категориальных признаков =================")

categorical\_features = train.select\_dtypes(include=['object']).columns.tolist()

if categorical\_features:

    categorical\_modes = train[categorical\_features].mode().iloc[0] if not train[categorical\_features].mode().empty else None

    categorical\_frequencies = train[categorical\_features].apply(lambda x: x.value\_counts().iloc[0] if not x.value\_counts().empty else 0)

    categorical\_stats = pd.DataFrame({"Mode": categorical\_modes, "Frequency": categorical\_frequencies})

    print(categorical\_stats)

else:

    print("Категориальные признаки отсутствуют.")

print("\n================= 6. Подготовка данных =================")

if not train.empty:

    train.fillna(train.median(), inplace=True)

    print("Пропущенные значения обработаны.")

else:

    print("Данные train отсутствуют.")

print("\n================= 7. Выбор признаков =================")

selected\_features = ["gravity", "ph"]

if all(feature in train.columns for feature in selected\_features):

    X\_selected = train[selected\_features].values

    y\_selected = train["target"].values

else:

    print("Ошибка: Выбранные признаки отсутствуют в данных.")

    exit()

print("\n================= 8. Нормализация данных =================")

scaler = StandardScaler()

X\_selected\_scaled = scaler.fit\_transform(X\_selected)

print("Нормализация выполнена.")

print("\n================= 9. Разделение данных на train/test =================")

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_selected\_scaled, y\_selected, test\_size=0.2, random\_state=42)

print("Разделение выполнено.")

print("\n================= 10. Обучение моделей =================")

models = {

    "KNN (собственная реализация)": KNN(k=5),

    "KNN": KNeighborsClassifier(n\_neighbors=5),

    "Logistic Regression": LogisticRegression(),

    "SVM": SVC(probability=True)

}

results = {}

for name, model in models.items():

    try:

        model.fit(X\_train, y\_train)

        y\_pred = model.predict(X\_val)

        if name == "KNN (собственная реализация)":

            y\_prob = np.zeros\_like(y\_pred, dtype=float)  # KNN не даёт вероятности

        else:

            y\_prob = model.predict\_proba(X\_val)[:, 1]

        acc = accuracy\_score(y\_val, y\_pred)

        prec = precision\_score(y\_val, y\_pred)

        rec = recall\_score(y\_val, y\_pred)

        f1 = f1\_score(y\_val, y\_pred)

        auc = roc\_auc\_score(y\_val, y\_prob) if name != "KNN (собственная реализация)" else "N/A"

        results[name] = [acc, prec, rec, f1, auc]

        print(f"\n{name}:\n Accuracy: {acc:.4f}\n Precision: {prec:.4f}\n Recall: {rec:.4f}\n F1: {f1:.4f}\n AUC: {auc}")

    except Exception as e:

        print(f"Ошибка при обучении {name}: {e}")

print("\n================= 11. Выбор лучшей модели =================")

if results:

    best\_model = max(results, key=lambda x: results[x][-1] if isinstance(results[x][-1], float) else results[x][-2])

    print(f"Лучшая модель: {best\_model}")

else:

    print("Не удалось выбрать лучшую модель.")

#print("\n================= 12. Визуализация границ классов =================")

def plot\_decision\_boundary(model, X, y, feature\_names):

    try:

        h = 0.02  # Шаг сетки

        x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

        y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

        xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 100),

                             np.linspace(y\_min, y\_max, 100))

        Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

        Z = Z.reshape(xx.shape)

        cmap\_light = ListedColormap(["#FFAAAA", "#AAFFAA"])

        cmap\_bold = ListedColormap(["#FF0000", "#AAFFAA"])

        plt.figure(figsize=(8, 6))

        plt.contourf(xx, yy, Z, cmap=cmap\_light, alpha=0.6)

        plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap\_bold, edgecolor="k", s=20)

        plt.xlabel(feature\_names[0])

        plt.ylabel(feature\_names[1])

        plt.title(f"Границы классов для {feature\_names[0]} и {feature\_names[1]}")

        plt.show()

    except Exception as e:

        print(f"Ошибка при построении границ классов: {e}")

knn\_selected = KNeighborsClassifier(n\_neighbors=5)

knn\_selected.fit(X\_selected\_scaled, y\_selected)

plot\_decision\_boundary(knn\_selected, X\_selected\_scaled, y\_selected, selected\_features)

Часть 2.

from sqlalchemy import create\_engine

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

from matplotlib.colors import ListedColormap

from collections import Counter

#  Реализация KNN с нуля

class KNN:

    def \_\_init\_\_(self, k=5):

        """ Инициализация модели с параметром k (количество ближайших соседей). """

        self.k = k

    def fit(self, X\_train, y\_train):

        """ Запоминаем обучающие данные. """

        self.X\_train = np.array(X\_train)

        self.y\_train = np.array(y\_train)

    def predict(self, X\_test):

        """ Для каждого объекта из тестового набора находим предсказанный класс. """

        predictions = [self.\_predict(x) for x in np.array(X\_test)]

        return np.array(predictions)

    def \_predict(self, x):

        """ Находим класс для одного объекта x. """

        distances = [self.\_euclidean\_distance(x, x\_train) for x\_train in self.X\_train]

        k\_indices = np.argsort(distances)[:self.k]

        k\_nearest\_labels = [self.y\_train[i] for i in k\_indices]

        most\_common = Counter(k\_nearest\_labels).most\_common(1)[0][0]

        return most\_common

    def \_euclidean\_distance(self, x1, x2):

        """ Вычисление евклидова расстояния между двумя точками. """

        return np.sqrt(np.sum((x1 - x2) \*\* 2))

# Данные для подключения

host = "povt-cluster.tstu.tver.ru"

port = 5432

user = "mpi"

password = "135a1"

database = "leonov"

# Функция подключения к базе данных

def connect\_to\_database():

    try:

        engine = create\_engine(f"postgresql://{user}:{password}@{host}:{port}/{database}?client\_encoding=utf8")

        print("Подключение успешно")

        return engine

    except Exception as e:

        print("Ошибка подключения:", e)

        exit()

# Функция загрузки полной базы данных

def load\_full\_data(engine):

    query = '''

    select

        k.\*,

        g."adi" as "группа крови",

        ilce."adi" as "район рождения",

        il."adi" as "город рождения",

        fak."adi" as "факультет",

        bol."adi" as "отделение",

        d."adi" as "курс",

        dao."vize" as "оценка за промежуточный экзамен",

        dao."final" as "оценка за финальный экзамен",

        dao."harf" as "оценка (буквенная)",

        yo."ucret" as "оплата за летнюю школу",

        case

            when k."cinsiyet" = true then 'мужчина'

            when k."cinsiyet" = false then 'женщина'

            else 'не указан'

        end as "пол"

    from

        "tkullanicilar" k

    left join

        "tkangruplari" g on k."kangrubu\_tkangruplariid" = g."id"

    left join

        "tilceler" ilce on k."dogumyeri\_tilcelerid" = ilce."id"

    left join

        "tiller" il on ilce."il\_tillerid" = il."id"

    left join

        "togrenciler" ogr on k."id" = ogr."ogrenci\_tkullanicilarid"

    left join

        "tbolumler" bol on ogr."bolum\_tbolumlerid" = bol."id"

    left join

        "tfakulteler" fak on bol."fakulte\_tfakultelerid" = fak."id"

    left join

        "tdersialanogrenciler" dao on k."id" = dao."ogrenci\_tkullanicilarid"

    left join

        "tdersler" d on dao."ders\_tderslerid" = d."id"

    left join

        "tyazokuluucretleri" yo on k."id" = yo."ogrenci\_tkullanicilarid"

    limit 10000;

    '''

    return pd.read\_sql\_query(query, engine)

# Подключение к базе

engine = connect\_to\_database()

df = load\_full\_data(engine)

print("\n================= 1. Загрузка данных =================")

print(f"Размер данных: {df.shape}")

print("\n================= 2. Информация о данных =================")

print(df.info())

print("\n================= 3. Проверка пропущенных значений =================")

missing\_values = df.isnull().sum()

print(missing\_values if missing\_values.sum() > 0 else "Пропущенные значения отсутствуют.")

print("\n================= 4. Анализ числовых признаков =================")

numerical\_features = df.select\_dtypes(include=['float64', 'int64']).columns.tolist()

numerical\_stats = df[numerical\_features].agg(['min', 'median', 'mean', 'max']).T

numerical\_stats["25%"] = df[numerical\_features].apply(lambda x: np.percentile(x.dropna(), 25))

numerical\_stats["75%"] = df[numerical\_features].apply(lambda x: np.percentile(x.dropna(), 75))

print(numerical\_stats)

print("\n================= 5. Анализ категориальных признаков =================")

categorical\_features = df.select\_dtypes(include=['object']).columns.tolist()

if categorical\_features:

    categorical\_modes = df[categorical\_features].mode().iloc[0]

    categorical\_frequencies = df[categorical\_features].apply(lambda x: x.value\_counts().iloc[0] if not x.value\_counts().empty else 'Нет данных')

    categorical\_stats = pd.DataFrame({"Mode": categorical\_modes, "Frequency": categorical\_frequencies})

    print(categorical\_stats)

else:

    print("Категориальные признаки отсутствуют.")

print("\n================= 6. Подготовка данных =================")

df.fillna(df.select\_dtypes(include=[np.number]).median(), inplace=True)

print("Пропущенные значения обработаны.")

# Преобразование целевой переменной в числовой формат (1 - мужчина, 0 - женщина)

df["пол"] = df["пол"].map({"мужчина": 1, "женщина": 0})

print("\n================= 7. Выбор признаков =================")

selected\_features = ["оценка за промежуточный экзамен", "оценка за финальный экзамен"]

if all(feature in df.columns for feature in selected\_features):

    X\_selected = df[selected\_features].values

    y\_selected = df["пол"].values

else:

    print("Ошибка: Выбранные признаки отсутствуют в данных.")

    exit()

print("\n================= 8. Нормализация данных =================")

scaler = StandardScaler()

X\_selected\_scaled = scaler.fit\_transform(X\_selected)

print("Нормализация выполнена.")

print("\n================= 9. Разделение данных на train/test =================")

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_selected\_scaled, y\_selected, test\_size=0.2, random\_state=42)

print("Разделение выполнено.")

print("\n================= 10. Обучение моделей =================")

models = {

    "KNN (собственная реализация)": KNN(k=5),

    "KNN": KNeighborsClassifier(n\_neighbors=5),

    "Logistic Regression": LogisticRegression(),

    "SVM": SVC(probability=True)

}

results = {}

for name, model in models.items():

    try:

        model.fit(X\_train, y\_train)

        y\_pred = model.predict(X\_val)

        if name == "KNN (собственная реализация)":

            y\_prob = np.zeros\_like(y\_pred, dtype=float)  # KNN не даёт вероятности

        else:

            y\_prob = model.predict\_proba(X\_val)[:, 1]

        acc = accuracy\_score(y\_val, y\_pred)

        prec = precision\_score(y\_val, y\_pred)

        rec = recall\_score(y\_val, y\_pred)

        f1 = f1\_score(y\_val, y\_pred)

        auc = roc\_auc\_score(y\_val, y\_prob) if name != "KNN (собственная реализация)" else "N/A"

        results[name] = [acc, prec, rec, f1, auc]

        print(f"\n{name}:\n Accuracy: {acc:.4f}\n Precision: {prec:.4f}\n Recall: {rec:.4f}\n F1: {f1:.4f}\n AUC: {auc}")

    except Exception as e:

        print(f"Ошибка при обучении {name}: {e}")

print("\n================= 11. Выбор лучшей модели =================")

if results:

    best\_model = max(results, key=lambda x: results[x][-1] if isinstance(results[x][-1], float) else results[x][-2])

    print(f"Лучшая модель: {best\_model}")

else:

    print("Не удалось выбрать лучшую модель.")

def plot\_decision\_boundary(model, X, y, feature\_names):

    try:

        h = 0.02  # Шаг сетки

        x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

        y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

        xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 100),

                             np.linspace(y\_min, y\_max, 100))

        Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

        Z = Z.reshape(xx.shape)

        cmap\_light = ListedColormap(["#FFAAAA", "#AAFFAA"])

        cmap\_bold = ListedColormap(["#FF0000", "#AAFFAA"])

        plt.figure(figsize=(8, 6))

        plt.contourf(xx, yy, Z, cmap=cmap\_light, alpha=0.6)

        plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap\_bold, edgecolor="k", s=20)

        plt.xlabel(feature\_names[0])

        plt.ylabel(feature\_names[1])

        plt.title(f"Границы классов для {feature\_names[0]} и {feature\_names[1]}")

        plt.show()

    except Exception as e:

        print(f"Ошибка при построении границ классов: {e}")

knn\_selected = KNeighborsClassifier(n\_neighbors=5)

knn\_selected.fit(X\_selected\_scaled, y\_selected)

plot\_decision\_boundary(knn\_selected, X\_selected\_scaled, y\_selected, selected\_features)