Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по практической работе №1**

По дисциплине: «Анализ больших данных»

Тема: «**Работа с библиотекой streamlit**»

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

Тверь 2025

**Оглавление**

[**Введение** 3](#_Toc191398103)

[**Цель:** 3](#_Toc191398104)

[**Формулировка задачи:** 3](#_Toc191398105)

[**Задача 1. Разбор данных из Kaggle** 3](#_Toc191398106)

[**Ссылка на репозиторий с программной реализацией:** 4](#_Toc191398107)

[**Описание проделанной работы** 5](#_Toc191398108)

[**Описание программы:** 5](#_Toc191398109)

[**Описание алгоритмов:** 5](#_Toc191398110)

[**Описание данных:** 8](#_Toc191398111)

[**Методы анализа данных и предобработки:** 8](#_Toc191398112)

[**Обучение и оценка моделей:** 8](#_Toc191398113)

[**Визуализация:** 8](#_Toc191398114)

[**Результаты** 10](#_Toc191398115)

[**Общее задание** 10](#_Toc191398116)

[**Описание базы данных – признаки, целевая переменная:** 10](#_Toc191398117)

[**Исследовательский анализ данных:** 11](#_Toc191398118)

[**Вывод по анализу:** 12](#_Toc191398119)

[**Визуализация результата:** 13](#_Toc191398120)

[**Критерии качества:** 15](#_Toc191398121)

[**Вывод по результату и критериям:** 15](#_Toc191398122)

[**Плюсы и минусы streamlit** 15](#_Toc191398123)

[**Веб интерфейс** 16](#_Toc191398124)

[**Вывод** 20](#_Toc191398125)

[**Использованные материалы** 21](#_Toc191398126)

[**Листинг кода** 22](#_Toc191398127)

# **Введение**

## **Цель:**

Разобраться с представлением отчетов в веб-версии с помощью библиотеки Streamlit. Реализовать отчет по одной из готовых лабораторных работ в формате веб-приложения.

## **Формулировка задачи:**

## **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?

**Вариант 9**

БД- Playground Series - Season 3, Episode 12

Ссылка на БД- https://www.kaggle.com/competitions/playground-series-s3e12/data?select=train.csv

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

GitHub- https://github.com/akkafe1ix/BigData\_Practice\_1

# **Описание проделанной работы**

## **Описание программы:**

Программа предназначена для анализа данных и машинного обучения. Реализована веб-версия отчета с использованием Streamlit. В приложении выполняются:

* загрузка и анализ данных;
* предварительная обработка;
* обучение моделей (KNN, логистическая регрессия, SVM);
* оценка качества моделей;
* визуализация границ решений.

Приложение включает несколько вкладок для удобного представления информации:

1. Об программе – описание проекта.
2. Датасет – представление загруженного набора данных.
3. EDA (Exploratory Data Analysis) – исследовательский анализ данных.
4. Результаты моделей – сравнительный анализ точности моделей.
5. Визуализация моделей – отображение границ решений.

## **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.

## **Описание данных:**

Программа работает с датасетом, содержащим числовые признаки:

* gravity – гравитация
* ph – кислотность
* osmo – осмотическое давление
* cond – проводимость
* urea – уровень мочевины
* calc – уровень кальция
* target – целевая переменная (бинарная классификация)

Данные загружаются из файлов train.csv и test.csv. Отображается информация о количестве строк, столбцов и размера датасета.

## **Методы анализа данных и предобработки:**

Перед обучением моделей выполняется анализ данных:

* Вычисление статистических характеристик (минимум, максимум, среднее, медиана, квартильные значения);
* Анализ категориальных признаков (если присутствуют);
* Масштабирование признаков с помощью StandardScaler.

## **Обучение и оценка моделей:**

В программе используются три модели машинного обучения:

1. KNN (K-Nearest Neighbors) – реализация метода ближайших соседей вручную и с использованием библиотеки sklearn.
2. Логистическая регрессия – базовая линейная модель классификации.
3. SVM (Support Vector Machine) – метод опорных векторов.

Для оценки моделей применяются метрики:

* Accuracy (точность);
* Precision (точность положительного класса);
* Recall (полнота);
* F1-score (среднее гармоническое precision и recall);
* ROC-AUC (качество разделения классов, если применимо).

На основе результатов вычисляется лучшая модель.

## **Визуализация:**

Для наглядного представления классификации строятся границы решений моделей. Используется цветовая карта ListedColormap для отображения принадлежности точек к классам.

Объекты данных отображаются в виде таблиц и диаграмм, что позволяет пользователю легко интерпретировать информацию. Вкладка "Датасет" показывает исходные данные в табличном виде с возможностью прокрутки и фильтрации. В "EDA" строятся гистограммы распределения признаков, боксплоты и корреляционные матрицы. Визуализация моделей включает интерактивные графики границ классификации, что упрощает понимание работы алгоритмов.

# **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):**

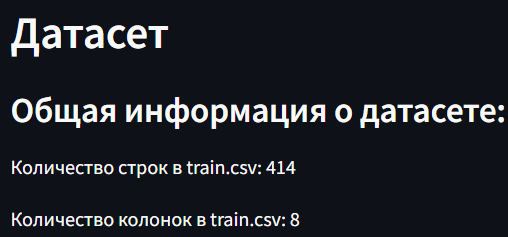
Целевая переменная **target** представляет собой бинарную классификацию:

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

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

### **Исследовательский анализ данных:**

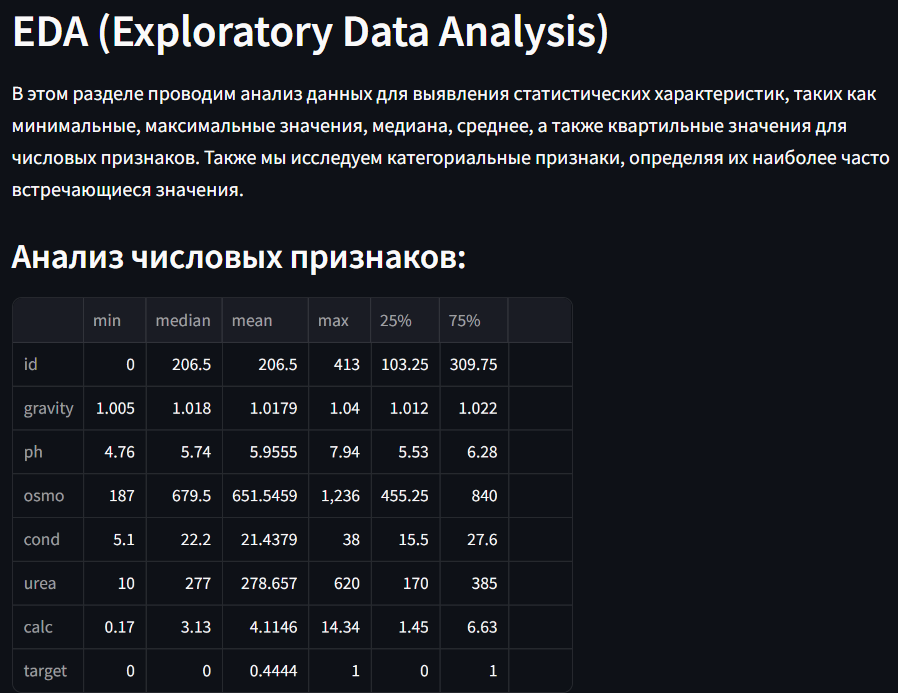
A. Сколько строк в датафрейме, сколько столбцов



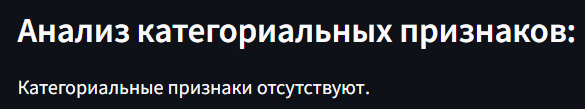
B. Сколько места занимает датафрейм в оперативной памяти



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.

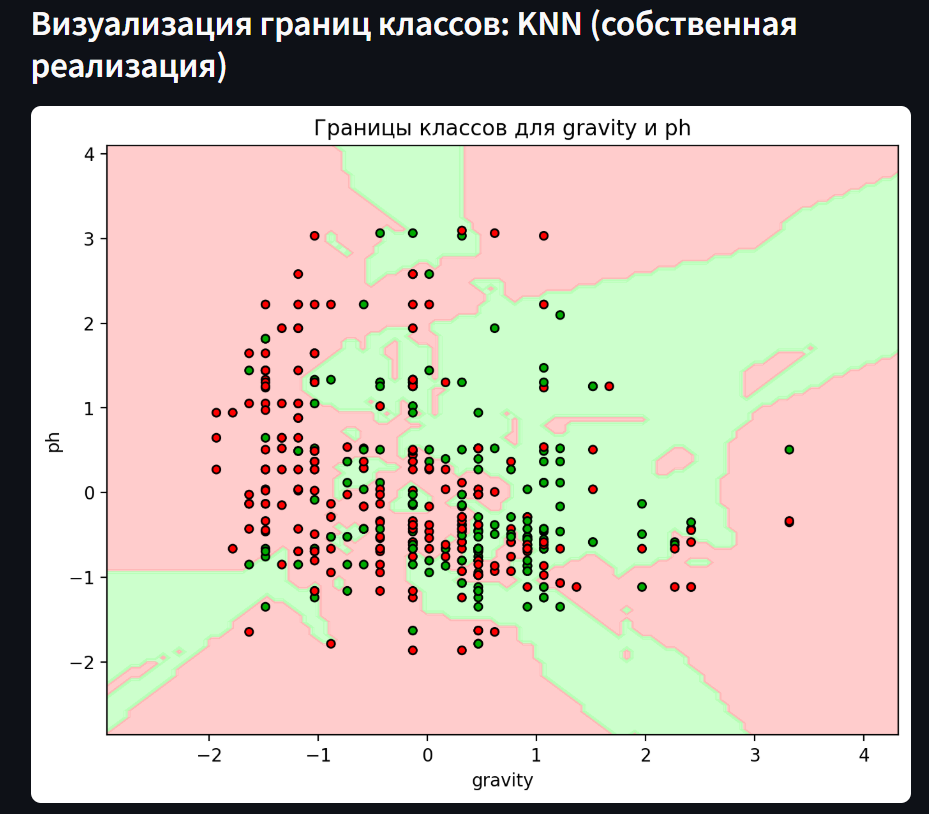
**Пропущенные значения:**

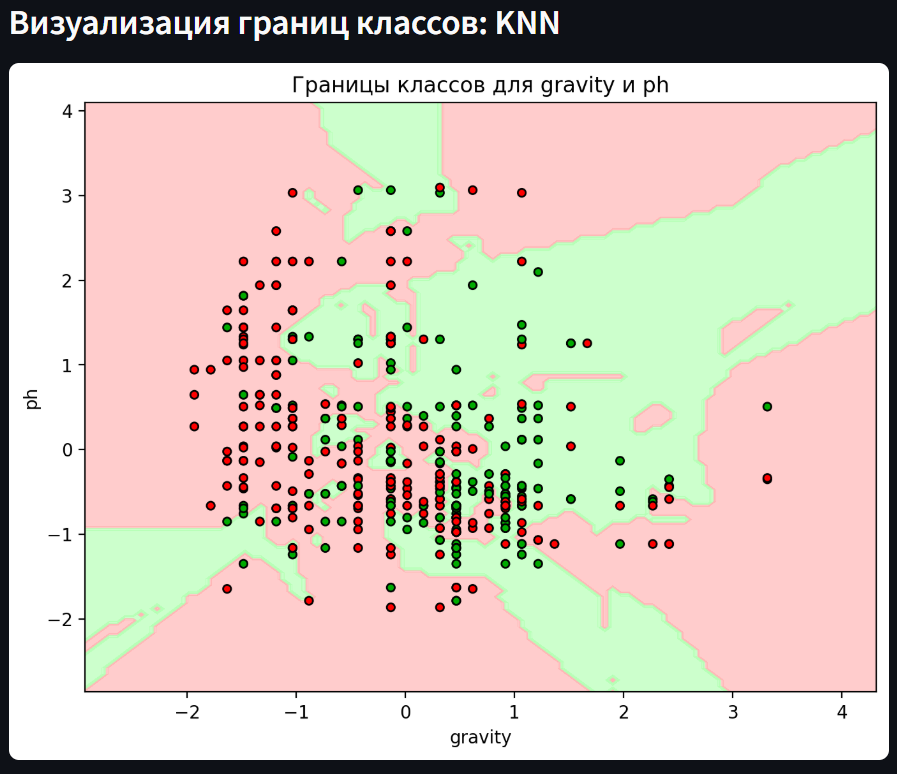
* **Пропущенные значения отсутствуют**, данные полностью заполнены.

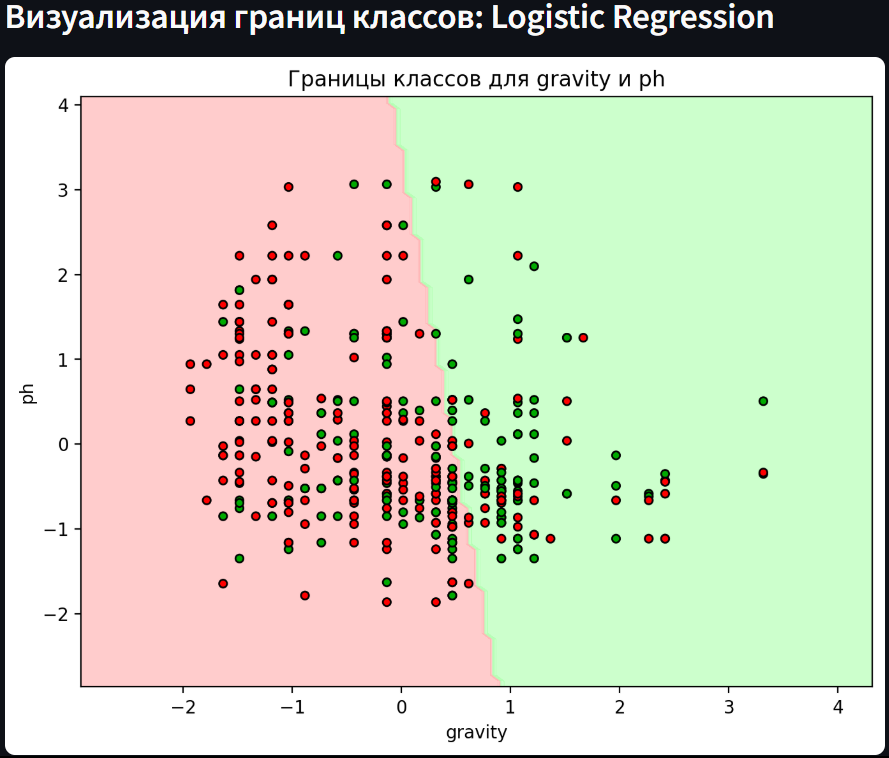
**Мода для целевой переменной:**

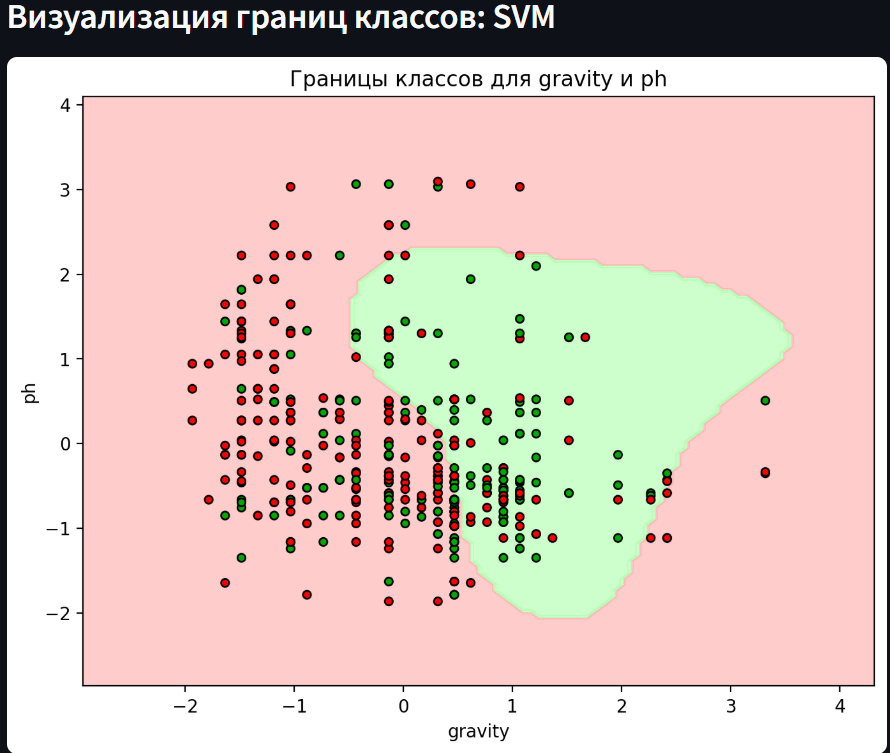
* Для целевой переменной **target** мода — **0**, что означает, что **класс 0 встречается чаще всего**.

### **Визуализация результата:**

****







### **Критерии качества:**



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

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

## **Плюсы и минусы streamlit**

Плюсы:

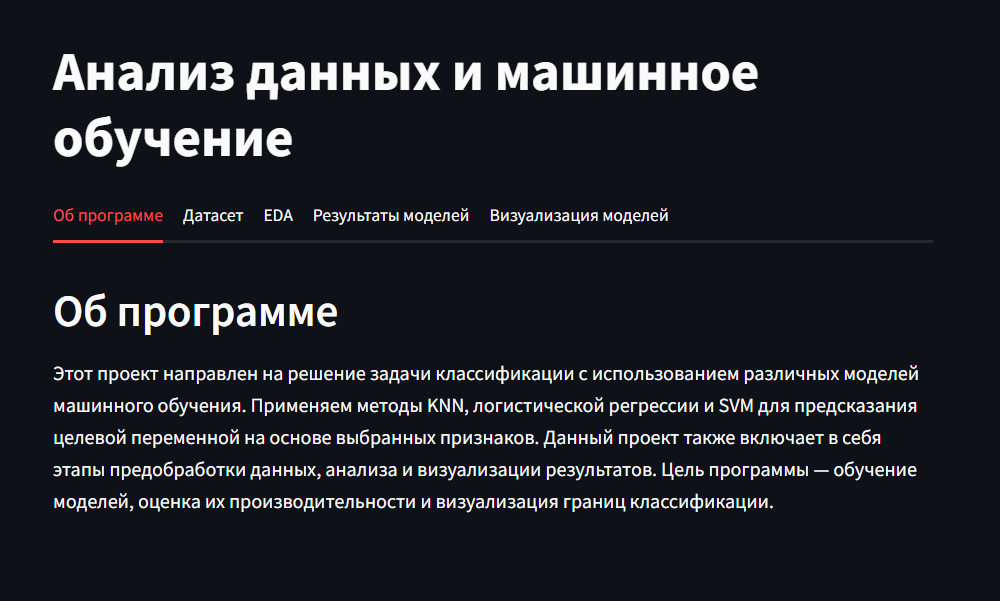
* Простота использования – позволяет быстро создавать веб-приложения без глубоких знаний веб-разработки.
* Быстрота разработки – код, написанный в Streamlit, минималистичен и легко модифицируется.
* Интерактивность – поддержка виджетов (st.slider, st.selectbox и др.) делает приложения удобными.
* Интеграция с Python – поддерживает работу с pandas, matplotlib, seaborn, sklearn и другими библиотеками.
* Развертывание без сложных настроек – можно запускать локально или развертывать в облаке с минимальными усилиями.
* Автоматическое обновление интерфейса – изменения в коде моментально отображаются в веб-приложении.

Минусы:

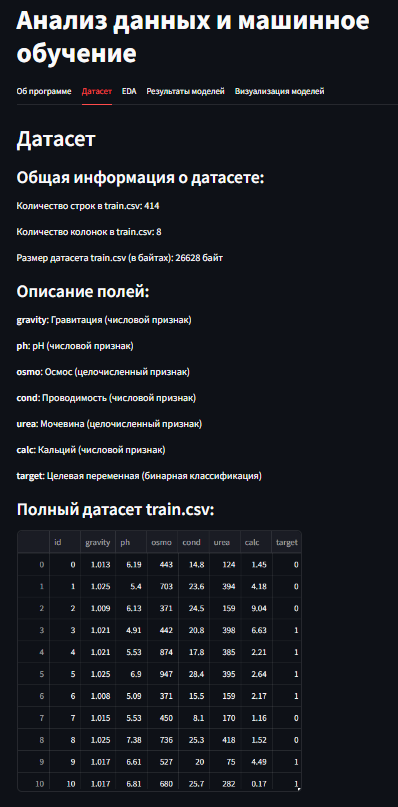
* Ограниченные возможности кастомизации интерфейса – сложнее настроить внешний вид по сравнению с Dash или Flask.
* Поддержка только Python – нельзя использовать JavaScript или другие языки для расширенной функциональности.
* Медленная обработка больших объемов данных – приложения могут тормозить при работе с большими датасетами.
* Не подходит для сложных веб-приложений – ограничен в плане создания сложных пользовательских взаимодействий.
* Зависимость от сервера – при развертывании в облаке требует постоянного соединения, что может увеличивать затраты.

## **Веб интерфейс**

Начальная вкладка:

****

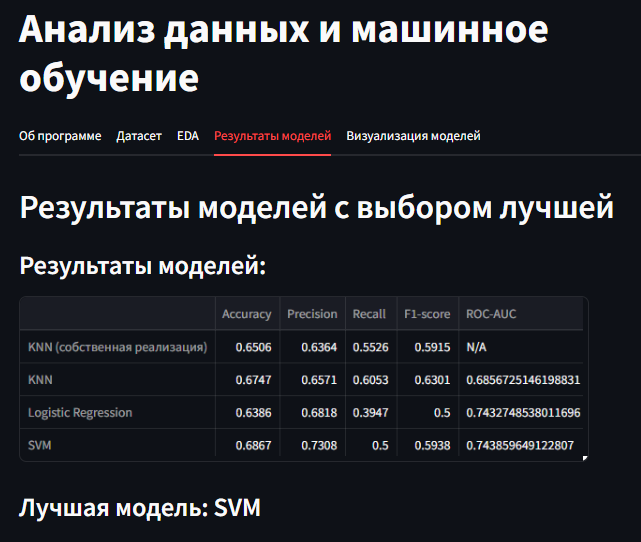
Описание датасета:

****

EDA:



Результаты моделей:



Визуализация моделей:



Далее идут другие интерпретации границ классов, а они не уместились в скриншот.

# **Вывод**

В результате работы реализовано веб-приложение, позволяющее загружать данные, анализировать их, обучать модели классификации, сравнивать их качество и визуализировать границы решений. Streamlit предоставляет удобный интерфейс для представления отчетов в интерактивной форме. Однако, несмотря на его удобство, при работе с большими объемами данных и сложными взаимодействиями могут возникнуть ограничения, требующие использования альтернативных решений.

# **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).
11. Основы Streamlit // Towards Data Science URL: https://towardsdatascience.com/streamlit (дата обращения: 24.02.2025).

# **Листинг кода**

import streamlit as st

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 collections import Counter

from matplotlib.colors import ListedColormap

# Реализация KNN с нуля

class KNN:

    def \_\_init\_\_(self, k=5):

        self.k = k

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

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

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

    def predict(self, X\_test):

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

        return np.array(predictions)

    def \_predict(self, 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):

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

# Структурируем страницу с вкладками

st.title("Анализ данных и машинное обучение")

# Вкладки

tab\_program, tab\_dataset, tab\_eda, tab\_results, tab\_visualization = st.tabs(

    ["Об программе", "Датасет", "EDA", "Результаты моделей", "Визуализация моделей"]

)

# Раздел "Об программе"

with tab\_program:

    st.header("Об программе")

    st.write("""

        Этот проект направлен на решение задачи классификации с использованием различных моделей машинного обучения.

        Применяем методы KNN, логистической регрессии и SVM для предсказания целевой переменной на основе выбранных признаков.

        Данный проект также включает в себя этапы предобработки данных, анализа и визуализации результатов.

        Цель программы — обучение моделей, оценка их производительности и визуализация границ классификации.

    """)

# Раздел "Датасет"

with tab\_dataset:

    st.header("Датасет")

    # Путь к файлам

    data\_path\_train = "C:\\Users\\Алексей\\Desktop\\8 Семестр\\(Корнеева) Анализ больших данных\\Практика\\Practic 1\\hello\\data\\train.csv"

    data\_path\_test = "C:\\Users\\Алексей\\Desktop\\8 Семестр\\(Корнеева) Анализ больших данных\\Практика\\Practic 1\\hello\\data\\test.csv"

    # Загрузка данных

    train = pd.read\_csv(data\_path\_train)

    test = pd.read\_csv(data\_path\_test)

    # Информация о датасете

    st.write("### Общая информация о датасете:")

    st.write(f"Количество строк в train.csv: {train.shape[0]}")

    st.write(f"Количество колонок в train.csv: {train.shape[1]}")

    st.write(f"Размер датасета train.csv (в байтах): {train.memory\_usage(deep=True).sum()} байт")

    # Описание каждого поля

    st.write("### Описание полей:")

    description = {

        "gravity": "Гравитация (числовой признак)",

        "ph": "pH (числовой признак)",

        "osmo": "Осмос (целочисленный признак)",

        "cond": "Проводимость (числовой признак)",

        "urea": "Мочевина (целочисленный признак)",

        "calc": "Кальций (числовой признак)",

        "target": "Целевая переменная (бинарная классификация)",

    }

    for col, desc in description.items():

        st.write(f"\*\*{col}\*\*: {desc}")

    # Отображение всего датасета

    st.write("### Полный датасет train.csv:")

    st.dataframe(train)

# Раздел "EDA"

with tab\_eda:

    st.header("EDA (Exploratory Data Analysis)")

    st.write("""

        В этом разделе проводим анализ данных для выявления статистических характеристик, таких как минимальные,

        максимальные значения, медиана, среднее, а также квартильные значения для числовых признаков.

        Также мы исследуем категориальные признаки, определяя их наиболее часто встречающиеся значения.

    """)

    st.write("### 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))

        st.dataframe(numerical\_stats)

    else:

        st.write("Числовые признаки отсутствуют.")

    st.write("### 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})

        st.dataframe(categorical\_stats)

    else:

        st.write("Категориальные признаки отсутствуют.")

# Раздел "Результаты моделей с выбором лучшей"

with tab\_results:

    st.header("Результаты моделей с выбором лучшей")

    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:

        st.error("Выбранные признаки отсутствуют в данных.")

        st.stop()

    scaler = StandardScaler()

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

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

    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():

        model.fit(X\_train, y\_train)

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

        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, model.predict\_proba(X\_val)[:, 1]) if hasattr(model, 'predict\_proba') else "N/A"

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

    st.write("### Результаты моделей:")

    results\_df = pd.DataFrame(results, index=["Accuracy", "Precision", "Recall", "F1-score", "ROC-AUC"])

    st.dataframe(results\_df.T)

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

    st.write(f"### Лучшая модель: {best\_model}")

# Раздел "Визуализация моделей"

with tab\_visualization:

    st.header("Визуализация моделей")

    st.write("""

        В этом разделе визуализируем границы решений для каждой модели.

        Границы разделения отображаются для выбранных признаков, что позволяет лучше понять, как каждая модель классифицирует данные.

    """)

    # Визуализация границ классов

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

        h = 0.02

        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", "#00AA00"])

        fig, ax = plt.subplots(figsize=(8, 6))

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

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

        ax.set\_xlabel(feature\_names[0])

        ax.set\_ylabel(feature\_names[1])

        ax.set\_title(f"Границы классов для {feature\_names[0]} и {feature\_names[1]}")

        st.pyplot(fig)

        plt.close()

    for name in models:

        st.write(f"### Визуализация границ классов: {name}")

        plot\_decision\_boundary(models[name], X\_selected\_scaled, y\_selected, selected\_features)