Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №4**

По дисциплине: «Анализ больших данных»

Тема: «Машинное обучение с учителем. Методы регрессии»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС - 21.06  Чесноков Д.С. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2025

Оглавление

[Введение 3](#_Toc192426502)

[Краткий опрос по теоретической часть 3](#_Toc192426503)

[Задание на лабораторную работу 3](#_Toc192426504)

[**Задача 1. Работа с данными kaggle** 3](#_Toc192426505)

[**Задача 1. Данные по вариантам** 4](#_Toc192426506)

[**Самостоятельная работа. Задача 2** 4](#_Toc192426507)

[Описание проделанной работы 5](#_Toc192426508)

[Теоретическая часть 5](#_Toc192426509)

[**База данных 1** 8](#_Toc192426510)

[Краткий вывод по работе 14](#_Toc192426511)

[Ссылки на используемые материалы. Документация 14](#_Toc192426512)

# **Введение**

# **Краткий опрос по теоретической часть**

В данном разделе необходимо ответить на вопросы.

1. Что такое регрессионный анализ?
2. Какую задачу решает линейная регрессия?
3. Какие преимущества имеет линейная регрессия?
4. Какое существует общее уравнение для линейной регрессии для двумерного пространства?
5. Какие входные и выходные параметры используются в модели множественной линейной регрессии?
6. Как работает алгоритм k-ближайших соседей для регрессии? В чем отличие работы от классификации
7. Чем отличается LASSO от линейной регрессии?
8. Чем отличается ElasticNET от линейной регрессии?
9. Какие метрики оценки качества алгоритмов регрессии Вы рассчитали в задании? По какой формуле они рассчитываются?
10. Что означает операция нормализации признаков, и чем она отличается от операции стандартизации?

# **Задание на лабораторную работу**

## **Задача 1. Работа с данными kaggle**

Ваша задача: построить минимум две регрессионные модели с использованием алгоритмов KNN, линейной регрессии, гребневой регрессии, LASSO или ElasticNet. Затем сравните их производительность, используя метрики, такие как MAE, MSE, RMSE, MAPE, R^2 на тестовых данных.

Обоснуйте свой выбор лучшей модели, учитывая точность прогнозирования и обобщающую способность на новых данных.

*Описание хода работы:*

1. Загрузить набор данных по варианту.

2. Провести разведочный анализ данных, ответив на следующие вопросы:

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

3. Подготовка датасета к построению моделей ML

a. Провести анализ и обработку пропусков (либо заменить, либо удалить)

b. Провести анализ и обработку выбросов (либо заменить, либо удалить)

c. Провести анализ и обработку категориальных переменных (сколько таких переменных, закодируйте категориальные переменные одним из методов (one hot encoding, mean target, frequence encoding).

d. Построить и проверить минимум 2 гипотезы на данных.

e. Разделить датасет на трейн и тест.

4. Обучить любые 2 алгоритма:

a. Knn

b. Linear regression

c. LASSO

d. ElasticNet

5. Оценить качество алгоритмов, выбрать самый оптимальный алгоритм.

6. Выгрузить и загрузить в программу натренированную модель данных. Использовать формат выгрузки библиотеки joblib.

## **Задача 1. Данные по вариантам**



## **Самостоятельная работа. Задача 2**

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

**Ссылка на репозиторий с программной реализацией**:

**Сложность: Medium**

* Реализовать общую часть задания
* Добавить нормализацию исходных данных, если необходимо
* Реализовать 2 алгоритма
* Реализовать выгрузку данных
* Реализовать хотя бы 1 алгоритм для самостоятельной части задания по Лабораторной 1 или выбранному набору данных

**Описание проделанной работы**

**Теоретическая часть**

1. **Что такое регрессионный анализ?**

Регрессионный анализ — это метод статистического моделирования, который изучает зависимость одной или нескольких зависимых (целевых) переменных от одной или нескольких независимых (предикторных) переменных. Его цель — предсказать значение целевой переменной на основе известных значений предикторов.

1. **Какую задачу решает линейная регрессия?**

Линейная регрессия решает задачу прогнозирования непрерывной числовой переменной, используя линейную зависимость между входными признаками (факторами) и целевой переменной.

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

* Простота реализации и интерпретации.
* Низкие вычислительные затраты.
* Возможность анализа значимости отдельных переменных.
* Хорошо работает, если зависимость между переменными действительно линейная.

4. Какое существует общее уравнение для линейной регрессии для двумерного пространства?

y - предсказанное значение

w0 - свободный коэффициент (intercept)

w1 - коэффициент наклона (вес)

x — входная переменная (признак).

5. Какие входные и выходные параметры используются в модели множественной линейной регрессии?

* Входные параметры: матрица признаков X, где каждая строка — объект, а столбцы — признаки.
* Выходные параметры: вектор целевой переменной y, содержащий прогнозируемые значения.

6. Как работает алгоритм k-ближайших соседей для регрессии? В чем отличие работы от классификации?

Алгоритм k-ближайших соседей (KNN) для регрессии:

* Выбирает k ближайших объектов из обучающей выборки по метрике расстояния (например, евклидово расстояние).
* Усредняет значения целевой переменной y среди этих соседей.
* Выдает среднее значение как предсказанный результат.
* Отличие от классификации: в классификации алгоритм KNN определяет наиболее частый класс среди соседей, а в регрессии вычисляет среднее значение целевой переменной.

7. Чем отличается LASSO от линейной регрессии?

* LASSO (Least Absolute Shrinkage and Selection Operator) — это линейная регрессия с L1-регуляризацией, которая добавляет штраф за модуль коэффициентов.
* LASSO автоматически выполняет отбор признаков, так как некоторые коэффициенты могут стать нулевыми, исключая нерелевантные признаки.

8. Чем отличается ElasticNet от линейной регрессии?

* ElasticNet сочетает L1-регуляризацию (LASSO) и L2-регуляризацию (Ridge), обеспечивая баланс между штрафами за большие коэффициенты и их обнулением.
* Лучше работает, когда у признаков есть сильная коллинеарность (зависимость между собой).

9. Какие метрики оценки качества алгоритмов регрессии Вы рассчитали в задании? По какой формуле они рассчитываются?

Метрики оценки качества регрессии:

**MAE (Mean Absolute Error) — средняя абсолютная ошибка**

**MSE (Mean Squared Error) — среднеквадратичная ошибка**

**RMSE (Root Mean Squared Error) — корень из среднеквадратичной ошибки**

**MAPE (Mean Absolute Percentage Error) — средняя абсолютная процентная ошибка**

R2 (коэффициент детерминации)

10. Что означает операция нормализации признаков, и чем она отличается от операции стандартизации?

Нормализация (Min-Max Scaling) — приведение значений признаков к диапазону [0,1] или [-1,1]:

Стандартизация (Z-Score Normalization) — приведение данных к нормальному распределению с нулевым средним и единичной дисперсией:

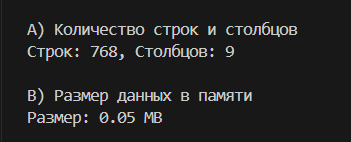
## **База данных 1**

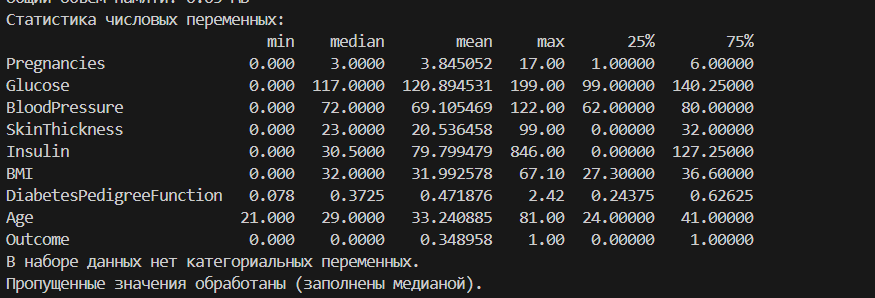
a. Описание базы данных

База данных представляет собой набор данных о диабете. Включает следующие признаки:

* Числовые признаки:
* pregnancies - беременности
* Glucose – глюкоза
* Bloodpressure - артериальное давление
* skin thickness – толщина кожи
* insulin – инсулин
* bmi ИМТ: Индекс массы тела
* DiabetesPedigreeFunction: Функция родословной диабета
* Age – возраст
* Целевая переменная: Outcome -диагноз (диабет/нет диабета).

b. Исследовательский анализ данных



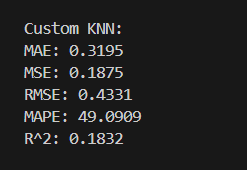


c. Выводы по анализу

* В данных есть пропущенные значения, которые были заменены на медианы.
* Выбросы обработаны путём усечения значений за 3 стандартных отклонения.
* Все признаки являются числовыми, следовательно, кодирование категориальных переменных не требуется.

Алгоритм 1: KNN

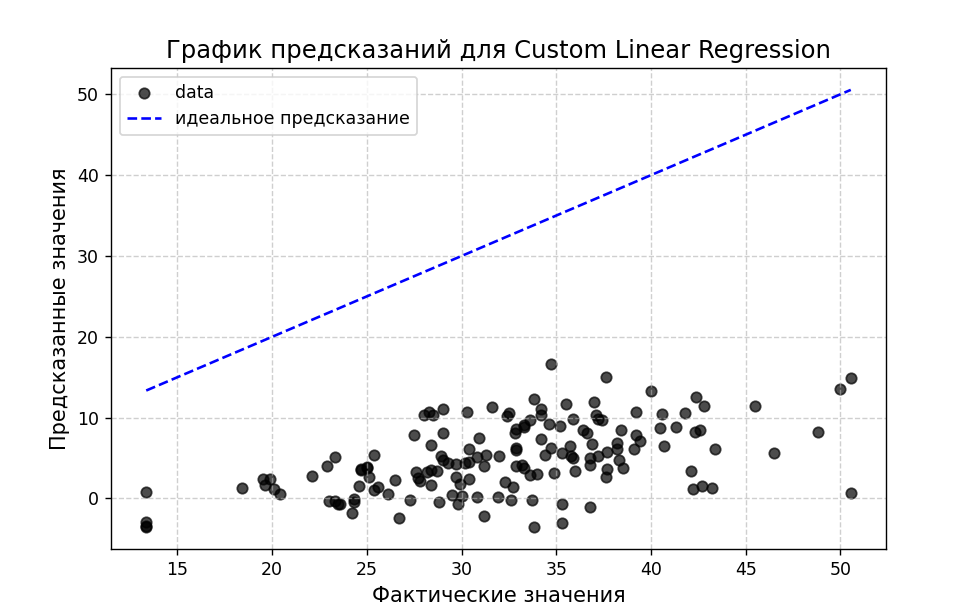
Критерии качества



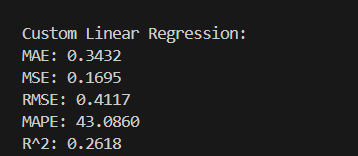
Алгоритм 2: Линейная регрессия

Визуализация результата

Target\_column – BMI feature\_column - Glucosse



Критерии качества;



## База данных 2

a. Описание базы данных – признаки, целевая переменная;

База данных: db\_housing

Таблица: Nashville\_Housing

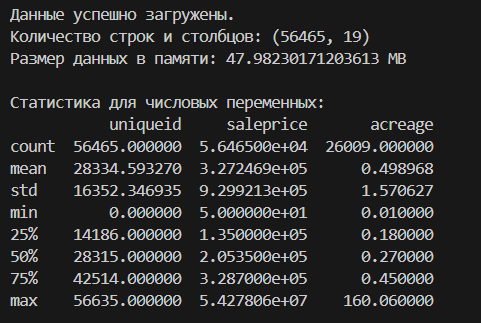
Количество записей: 56465 строк, 19 столбцов

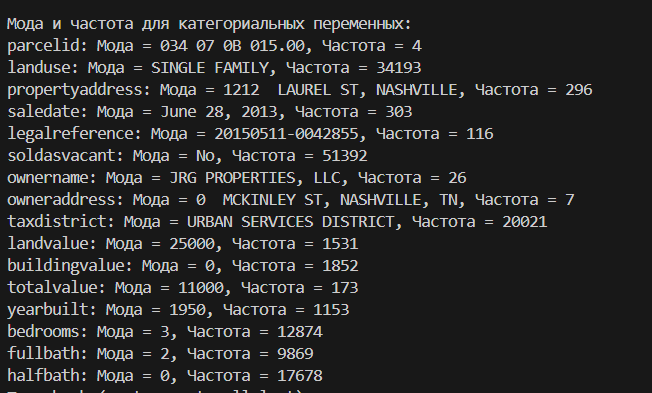
Целевая переменная: saleprice

Признаки в базе данных (с указанием типа):

* uniqueid (Числовой): Уникальный идентификатор записи
* parcelid (Числовой): Идентификатор земельного участка
* landuse (Категориальный): Тип использования земли
* propertyaddress (Категориальный): Адрес недвижимости
* saledate (Категориальный): Дата продажи
* eference: Юридическая ссылка
* soldasvacant (Категориальный): Продана как пустая?
* ownername (Категориальный): Имя владельца
* owneraddress (Категориальный): Адрес владельца
* acreage (Числовой): Площадь участка (в акрах)
* taxdistrict (Категориальный): Налоговый округ
* landvalue (Числовой): Оценочная стоимость земли
* buildingvalue (Числовой): Оценочная стоимость здания
* totalvalue (Числовой): Общая стоимость недвижимости
* yearbuilt (Числовой): Год постройки
* bedrooms (Числовой): Количество спален
* fullbath (Числовой): Количество ванных комнат
* halfbath (Числовой): Количество половинных ванных комнат
* saleprice (Числовой, целевая переменная): Цена продажи

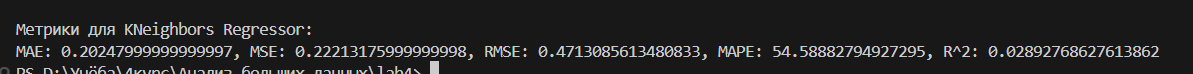
b. Исследовательский анализ данных (поставить вопросы на изучение по аналогии с заданием 1 и ответить на них);



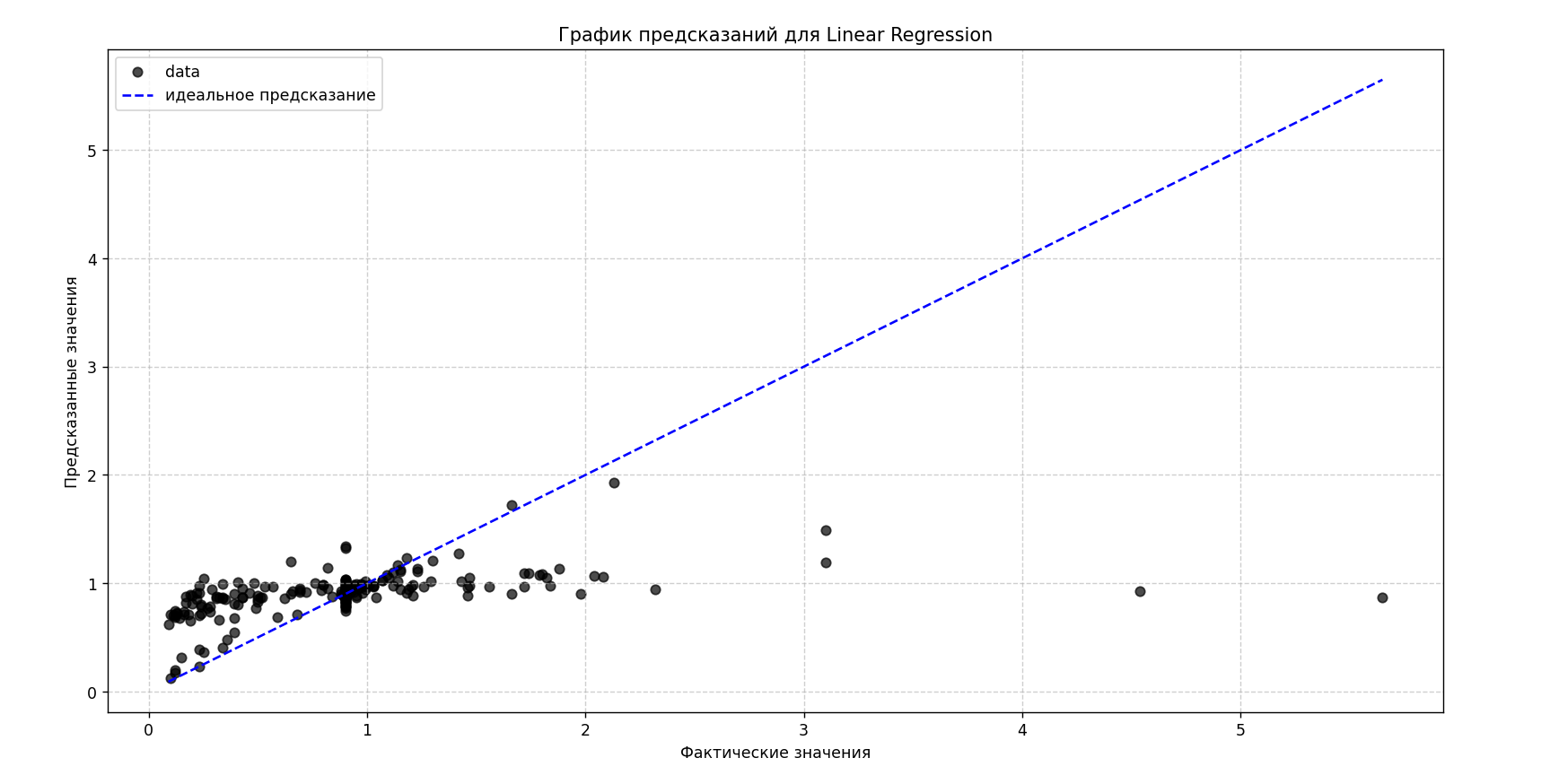


Выводы

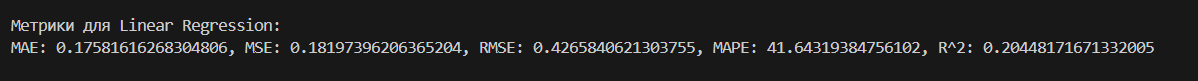
Алгоритм KNN;



**Алгоритм Линейная регрессия**



i. Критерии качества;



# **Краткий вывод по работе**

В ходе выполнения лабораторной работы была проведена полная обработка набора данных, включающая разведочный анализ, предобработку и построение нескольких регрессионных моделей.

Основные этапы работы:

1. Разведочный анализ данных
2. Предобработка данных
3. Обучение моделей

В результате работы была выбрана оптимальная модель на основе ее качества предсказаний и устойчивости к новым данным. Подход включал систематический анализ данных, предобработку и тестирование различных методов регрессии, что позволило получить наиболее эффективное решение для задачи прогнозирования.

# **Ссылки на используемые материалы. Документация**

1. Клиент-серверные приложения Сокеты. Создание клиента // METANIT.COM URL:https://metanit.com/python/network/1.1.php (дата обращения: 15.01.2025).
2. Python Documentation contents // python.org URL: https://docs.python.org/3/contents.html (дата обращения: 15.01.2025).
3. Визуализация данных с matplotlib // NSU Programming URL: https://nsu-programming.github.io/textbook/python/plotting.html (дата обращения: 15.01.2025).
4. Метод K-ближайших соседей (KNN). Принцип работы, разновидности и реализация с нуля на Python // habr.com URL: https://habr.com/ru/articles/801885/ (дата обращения: 10.02.2025).

**Листинг кода**

import pandas as pd

import numpy as np

from sklearn.discriminant\_analysis import StandardScaler

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

from sklearn.preprocessing import OneHotEncoder

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import ttest\_ind, pearsonr

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

from sklearn.preprocessing import OneHotEncoder

from sklearn.neighbors import KNeighborsRegressor

from sklearn.linear\_model import LinearRegression, Lasso, ElasticNet

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from scipy.stats import ttest\_ind, pearsonr

import joblib

def save\_model(model, filename):

    """Сохраняет модель в файл с использованием joblib."""

    joblib.dump(model, filename)

    print(f"Модель сохранена в {filename}")

def load\_model(filename):

    """Загружает модель из файла с использованием joblib."""

    model = joblib.load(filename)

    print(f"Модель загружена из {filename}")

    return model

def load\_data(file\_path):

    """Загружает данные из CSV-файла."""

    return pd.read\_csv(file\_path)

def data\_overview(df):

    """Выводит информацию о количестве строк и столбцов, а также о размере в памяти."""

    num\_rows, num\_cols = df.shape

    memory\_usage = df.memory\_usage(deep=True).sum() / 1024 \*\* 2  # В мегабайтах

    print(f"Количество строк: {num\_rows}")

    print(f"Количество столбцов: {num\_cols}")

    print(f"Общий объем памяти: {memory\_usage:.2f} MB")

def numeric\_summary(df):

    """Выводит статистики для числовых переменных (мин, медиана, среднее, макс, персентили 25 и 75)."""

    summary = df.describe(percentiles=[0.25, 0.75]).T[['min', '50%', 'mean', 'max', '25%', '75%']]

    summary.rename(columns={'50%': 'median'}, inplace=True)

    print("Статистика числовых переменных:")

    print(summary)

def categorical\_summary(df):

    """Выводит моду и количество ее повторений для категориальных переменных."""

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

    if len(categorical\_cols) == 0:

        print("В наборе данных нет категориальных переменных.")

    else:

        for col in categorical\_cols:

            mode\_value = df[col].mode()[0]

            mode\_count = df[col].value\_counts().iloc[0]

            print(f"Переменная: {col}")

            print(f"Мода: {mode\_value}, встречается {mode\_count} раз\n")

def handle\_missing\_values(df):

    """Обрабатывает пропущенные значения (заполняет медианой)."""

    df.fillna(df.median(numeric\_only=True), inplace=True)

    print("Пропущенные значения обработаны (заполнены медианой).")

def handle\_outliers(df):

    """Обрабатывает выбросы с использованием межквартильного размаха (IQR)."""

    for col in df.select\_dtypes(include=[np.number]).columns:

        Q1 = df[col].quantile(0.25)

        Q3 = df[col].quantile(0.75)

        IQR = Q3 - Q1

        lower\_bound = Q1 - 1.5 \* IQR

        upper\_bound = Q3 + 1.5 \* IQR

        df[col] = np.clip(df[col], lower\_bound, upper\_bound)

    print("Выбросы обработаны с использованием метода IQR.")

def encode\_categorical(df):

    """Кодирует категориальные переменные с помощью One-Hot Encoding."""

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

    if len(categorical\_cols) > 0:

        df = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

        print("Категориальные переменные закодированы методом One-Hot Encoding.")

    return df

def test\_hypotheses(df):

    """Проверяет две гипотезы на данных."""

    print("\nГипотеза 1: Средний уровень глюкозы у диабетиков выше, чем у здоровых людей.")

    diabetic = df[df['Outcome'] == 1]['Glucose']

    non\_diabetic = df[df['Outcome'] == 0]['Glucose']

    t\_stat, p\_value = ttest\_ind(diabetic, non\_diabetic)

    print(f"t-статистика: {t\_stat:.2f}, p-значение: {p\_value:.5f}")

    print("Гипотеза отвергается" if p\_value < 0.05 else "Гипотеза не отвергается")

    print("\nГипотеза 2: Корреляция между BMI и уровнем глюкозы положительная.")

    correlation, p\_value = pearsonr(df['BMI'], df['Glucose'])

    print(f"Коэффициент корреляции: {correlation:.2f}, p-значение: {p\_value:.5f}")

    print("Корреляция значима" if p\_value < 0.05 else "Корреляция не значима")

def split\_data(df, target\_column):

    X = df.drop(columns=[target\_column])

    y = df[target\_column].values

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

    scaler = StandardScaler()

    X\_train = scaler.fit\_transform(X\_train)

    X\_test = scaler.transform(X\_test)  # Применяем тот же scaler, что был обучен на X\_train

    return np.array(X\_train), np.array(X\_test), np.array(y\_train), np.array(y\_test), list(X.columns)

class CustomKNNRegressor:

    """Реализация алгоритма KNN для регрессии."""

    def \_\_init\_\_(self, n\_neighbors=5):

        self.n\_neighbors = n\_neighbors

    def fit(self, X, y):

        self.X\_train = np.array(X)

        self.y\_train = np.array(y)

    def predict(self, X):

        X = np.array(X)

        y\_pred = []

        for x in X:

            distances = np.linalg.norm(self.X\_train - x, axis=1)

            neighbors\_idx = np.argsort(distances)[:self.n\_neighbors]

            y\_pred.append(np.mean(self.y\_train[neighbors\_idx]))

        return np.array(y\_pred)

class CustomLinearRegression:

    def \_\_init\_\_(self, learning\_rate=1e-3, n\_iterations=5000, l2\_penalty=0.1, clip\_value=1.0):

        self.learning\_rate = learning\_rate

        self.n\_iterations = n\_iterations

        self.l2\_penalty = l2\_penalty  # Регуляризация L2

        self.clip\_value = clip\_value  # Ограничение градиента

    def fit(self, X, y):

        X = np.array(X)

        y = np.array(y)

        X = np.c\_[np.ones(X.shape[0]), X]  # Добавляем столбец единиц для свободного члена

        self.theta = np.zeros(X.shape[1])

        for i in range(self.n\_iterations):

            y\_pred = X @ self.theta

            error = y - y\_pred

            # Вычисление градиента с учетом L2-регуляризации

            gradients = (-2 / X.shape[0]) \* (X.T @ error) + (self.l2\_penalty / X.shape[0]) \* self.theta

            gradients = np.clip(gradients, -self.clip\_value, self.clip\_value)  # Ограничение градиента

            self.theta -= self.learning\_rate \* gradients

            # Проверка на NaN и бесконечность

            if np.isnan(self.theta).any() or np.isinf(self.theta).any():

                print(f"Градиентный спуск не сходится, остановка обучения на {i}-й итерации.")

                break

    def predict(self, X):

        X = np.array(X)

        X = np.c\_[np.ones(X.shape[0]), X]

        return X @ self.theta

def train\_models(X\_train, y\_train):

    """Обучает кастомные модели KNN и линейной регрессии."""

    models = {

        'Custom KNN': CustomKNNRegressor(n\_neighbors=5),

        'Custom Linear Regression': CustomLinearRegression()

    }

    trained\_models = {}

    for name, model in models.items():

        model.fit(X\_train, y\_train)

        trained\_models[name] = model

    return trained\_models

def evaluate\_models(models, X\_test, y\_test):

    """Оценивает модели с использованием метрик MAE, MSE, RMSE, MAPE и R^2."""

    results = {}

    for name, model in models.items():

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

        mae = mean\_absolute\_error(y\_test, y\_pred)

        mse = mean\_squared\_error(y\_test, y\_pred)

        rmse = np.sqrt(mse)

        # Убираем нулевые значения перед расчетом MAPE

        non\_zero\_mask = y\_test != 0

        if np.any(non\_zero\_mask):

            mape = np.mean(np.abs((y\_test[non\_zero\_mask] - y\_pred[non\_zero\_mask]) / y\_test[non\_zero\_mask])) \* 100

        else:

            mape = np.nan  # Если все значения нулевые, выставляем NaN

        r2 = r2\_score(y\_test, y\_pred)

        results[name] = {'MAE': mae, 'MSE': mse, 'RMSE': rmse, 'MAPE': mape, 'R^2': r2}

    return results

def plot\_predictions(model, X\_test, y\_test, model\_name="Linear Regression"):

    """

    Визуализация предсказанных значений относительно фактических.

    """

    # Предсказание

    y\_pred = model.predict(X\_test)

    # Построение графика

    plt.figure(figsize=(8, 5))

    plt.scatter(y\_test, y\_pred, color='black', label="data", alpha=0.7)

    plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', label="идеальное предсказание")

    plt.xlabel("Фактические значения", fontsize=12)

    plt.ylabel("Предсказанные значения", fontsize=12)

    plt.title(f"График предсказаний для {model\_name}", fontsize=14)

    plt.legend()

    plt.grid(True, linestyle="--", alpha=0.6)

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    file\_path = "diabetes.csv"  # Укажите путь к файлу, если он не в текущей директории

    df = load\_data(file\_path)

    df\_columns = df.columns.tolist()

    print(df\_columns)

    data\_overview(df)

    numeric\_summary(df)

    categorical\_summary(df)

    handle\_missing\_values(df)

    handle\_outliers(df)

    df = encode\_categorical(df)

    target\_column = "BMI"  # Целевая переменная

    feature\_column = "Glucosse"  # Признак для визуализации

    X\_train, X\_test, y\_train, y\_test, feature\_names = split\_data(df, target\_column)

    test\_hypotheses(df)

    print(f"\n")

    trained\_models = train\_models(X\_train, y\_train)

    model\_results = evaluate\_models(trained\_models, X\_test, y\_test)

    for model, metrics in model\_results.items():

        print(f"\n{model}:")

        for metric, value in metrics.items():

            print(f"{metric}: {value:.4f}")

    # Выводим список доступных признаков

    print("Доступные признаки:", feature\_names)

    model = trained\_models["Custom Linear Regression"]  # Берем именно обученную модель

    # Вызов функции

    plot\_predictions(model, X\_test, y\_test, "Custom Linear Regression")

    # Сохранение модели

    save\_model(trained\_models["Custom Linear Regression"], "custom\_linear\_regression.pkl")

    # Загрузка модели

    #loaded\_model = load\_model("custom\_linear\_regression.pkl")

    # Проверка работы загруженной модели

    #y\_pred = loaded\_model.predict(X\_test)

    # Визуализация предсказаний загруженной модели

    #plot\_predictions(loaded\_model, X\_test, y\_test, "Loaded Custom Linear Regression")