

Регрессионные и классификационные подходы к анализу диабетических данных в google colab

Неретин Андрей Сергеевич

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается процесс построения регрессионной и классификационной моделей на одном наборе данных. В качестве примера используется набор данных о диабете, содержащий медицинские показатели пациентов. Основное внимание уделяется этапам подготовки данных, выбору целевых переменных для каждой модели, а также применению алгоритмов машинного обучения. Для регрессионной модели используется метод линейной регрессии, позволяющий предсказывать количественные значения, такие как уровень глюкозы. Для классификационной модели применяется логистическая регрессия, направленная на прогнозирование наличия или отсутствия диабета у пациента. В статье также представлены методы оценки качества моделей, такие как среднеквадратичная ошибка (MSE) для регрессии и точность (accuracy) для классификации. Работа подчеркивает важность анализа данных и выбора подходящих алгоритмов для решения задач различного типа на одном наборе данных.

Ключевые слова: Регрессионные модели, классификационные модели, данные, среднеквадратичная ошибка, точность, анализ данных.

Regression and classification approaches to the analysis of diabetic data in google colab

Neretin Andrey Sergeevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the process of building regression and classification models on a single data set. As an example, a diabetes dataset containing the medical records of patients is used. The main focus is on the stages of data preparation, the selection of target variables for each model, as well as the application of machine learning algorithms. The regression model uses the linear regression method, which allows predicting quantitative values such as glucose levels. The classification model uses logistic regression aimed at predicting the presence or absence of diabetes in a patient. The article also presents methods for evaluating the quality of models, such as mean square error (MSE) for regression and accuracy for classification. The work

highlights the importance of data analysis and the selection of suitable algorithms for solving various types of problems on a single data set.

Keywords: Regression models, classification models, data, root-mean-square error, accuracy, data analysis.

1 Введение

1.1 Актуальность

Актуальность темы построения регрессионной и классификационной моделей на одном наборе данных обусловлена растущей значимостью анализа данных в различных областях, включая медицину, экономику и социальные науки. В условиях быстрого увеличения объемов данных и их разнообразия, умение эффективно применять методы машинного обучения становится ключевым навыком для специалистов. Использование одного набора данных для построения как регрессионной, так и классификационной модели позволяет не только оптимизировать процесс анализа, но и выявить взаимосвязи между количественными и категориальными переменными. Это особенно важно в медицинских исследованиях, где необходимо предсказывать как количественные показатели (например, уровень глюкозы), так и бинарные исходы (например, наличие или отсутствие диабета).

1.2 Обзор исследований

Н. А. Моисеева, Т. А. Полякова провели регрессионный анализ данных в технических исследованиях [1], Регрессионный анализ как средство изучения зависимости между переменными продемонстрировали Б.Ж.Мамуров, Ж. Ж. Абдуллаев [2], А. Е. Сенникова, Н. Х. Ворокова осуществили регрессионный анализ влияния объема и структуры основных фондов на эффективность сельскохозяйственного производства [3], А.Н.Калашников, М. Г. Тиндова, И. М. Кублин провели исследование социально-экономического положения региона методами факторного и регрессионного анализа[4], оценку факторов эксплуатационной технологичности машин методом регрессионного анализа осуществили В. Н. Шиловский, И.Г.Скобцов, Д. Г. Конанов [5].

1.3 Цель исследования

Целью исследования является разработка и сравнение регрессионной и классификационной моделей на диабетическом наборе данных, что позволит выявить ключевые взаимосвязи между различными медицинскими показателями и их влиянием на предсказание уровня глюкозы и наличия диабета. Исследование направлено на оптимизацию процесса анализа данных, а также на оценку эффективности различных алгоритмов машинного обучения в контексте решения задач регрессии и классификации.

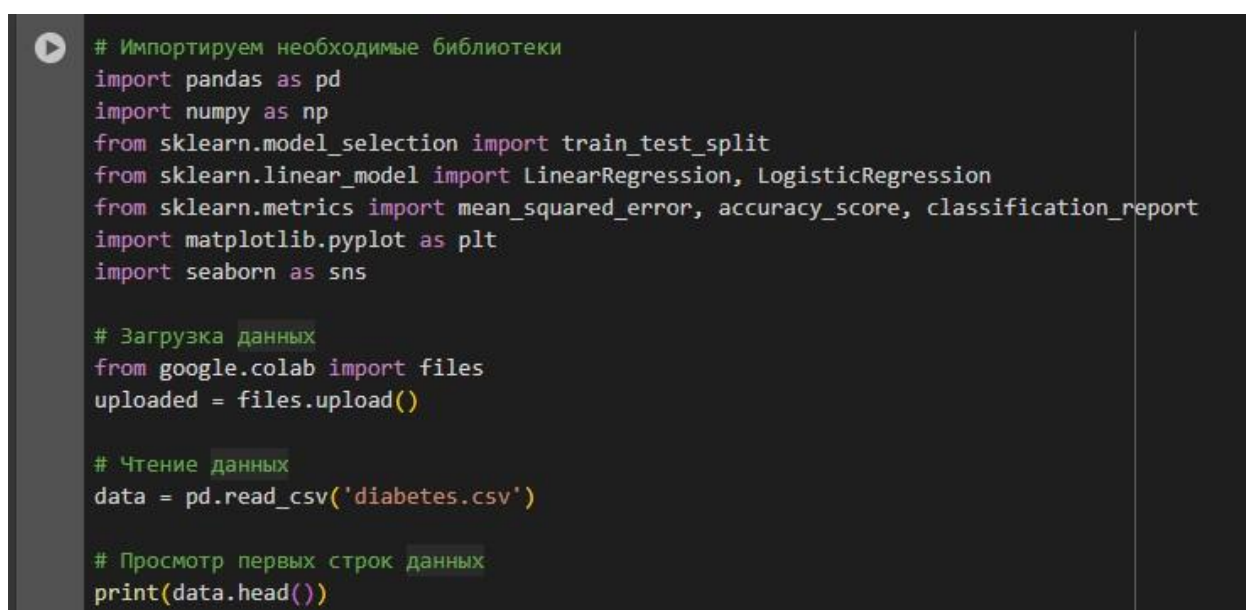
Материалы и методы

Для анализа используется набор данных diabetes.csv, который содержит информацию о пациентах с диабетом. Данные включают такие параметры, как

уровень глюкозы, индекс массы тела (ИМТ), возраст, артериальное давление и другие клинические показатели. Набор данных также содержит целевую переменную, которая может быть использована для классификации (например, наличие или отсутствие диабета) и для регрессии (например, предсказание уровня инсулина). Ссылка на данные: <https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset?resource=download>

3.Результаты и дискуссии

Для того чтобы начать работать с данными, требуется импортировать необходимые библиотеки, загрузить данные в проект google colab и просмотреть строки данных (рис.1).



```
# Импортируем необходимые библиотеки
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Загрузка данных
from google.colab import files
uploaded = files.upload()

# Чтение данных
data = pd.read_csv('diabetes.csv')

# Просмотр первых строк данных
print(data.head())
```

Рисунок 1. Импорт необходимых библиотек, загрузка данных в проект google colab и просмотр строк данных

После успешной загрузки данных, выведет окно с первыми пятью строками (рис.2).

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
	DiabetesPedigreeFunction	Age	Outcome				
0	0.627	50	1				
1	0.351	31	0				
2	0.672	32	1				
3	0.167	21	0				
4	2.288	33	1				

Рисунок 2. Окно с первыми пятью строками

Чтобы понимать названия столбцов, необходимо переименовать их на русский язык и проверить результат (рис.3).

```
[5] # Переименование столбцов на русский
data.rename(columns={
    'Pregnancies': 'Беременности',
    'Glucose': 'Глюкоза',
    'BloodPressure': 'Давление',
    'SkinThickness': 'Толщина_кожи',
    'Insulin': 'Инсулин',
    'BMI': 'ИМТ',
    'DiabetesPedigreeFunction': 'Наследственность',
    'Age': 'Возраст',
    'Outcome': 'Результат'
}, inplace=True)

# Проверим результат
print(data.head())
```

	Беременности	Глюкоза	Давление	Толщина_кожи	Инсулин	ИМТ	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
	Наследственность	Возраст	Результат				
0	0.627	50	1				
1	0.351	31	0				
2	0.672	32	1				
3	0.167	21	0				
4	2.288	33	1				

Рисунок 3. Переименование столбцов

Далее нужно понять для чего нужны и в чем измеряются столбцы в датасете:

1. Беременности:

Описание: Количество беременностей у пациента.

Единицы измерения: Целое число (количество).

2. Глюкоза:

Описание: Уровень глюкозы в крови после еды (в плазме).

Единицы измерения: Миллиграммы на децилитр (mg/dL).

3. Давление:

Описание: Диастолическое артериальное давление (нижнее значение).

Единицы измерения: Миллиметры ртутного столба (mmHg).

4. Толщина_кожи:

Описание: Толщина кожной складки на трицепсе. Используется как индикатор подкожного жира.

Единицы измерения: Миллиметры (mm).

5. Инсулин:

Описание: Уровень инсулина в сыворотке крови.

Единицы измерения: Микроединицы на миллилитр ($\mu\text{U/mL}$).

6. ИМТ:

Описание: Индекс массы тела (Body Mass Index), рассчитывается как вес (в кг) делённый на квадрат роста (в метрах).

Единицы измерения: Безразмерная величина.

7. Наследственность:

Описание: Функция родословной диабета, которая оценивает вероятность диабета на основе семейной истории.

Единицы измерения: Безразмерная величина (коэффициент).

8. Возраст:

Описание: Возраст пациента.

Единицы измерения: Годы.

9. Результат:

Описание: Целевой признак, указывающий на наличие диабета.

0: Диабета нет.

1: Диабет диагностирован.

Единицы измерения: Бинарное значение (0 или 1).

Следующим шагом проверим информацию о данных, проверим наличие пропущенных значений, сделаем статистическое описание данных и визуализируем корреляцию между признаками (рис.4).

```
# Проверим информацию о данных
print(data.info())

# Проверим наличие пропущенных значений
print(data.isnull().sum())

# Статистическое описание данных
print(data.describe())

# Визуализация корреляции между признаками
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Корреляция признаков')
plt.show()
```

Рисунок 4. Код для проверки информации о данных, проверки наличия пропущенных значений, статистическое описание данных и визуализация корреляции между признаками

После выполнения команды выведет следующее (рис.5).

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Беременности          768 non-null    int64  
1   Глюкоза               768 non-null    int64  
2   Давление              768 non-null    int64  
3   Толщина_кожи         768 non-null    int64  
4   Инсулин               768 non-null    int64  
5   ИМТ                   768 non-null    float64 
6   Наследственность     768 non-null    float64 
7   Возраст               768 non-null    int64  
8   Результат             768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
Беременности          0
Глюкоза               0
Давление              0
Толщина_кожи         0
Инсулин               0
ИМТ                   0
Наследственность     0
Возраст               0
Результат             0
dtype: int64

      Беременности      Глюкоза      Давление      Толщина_кожи      Инсулин  \
count      768.000000    768.000000    768.000000    768.000000    768.000000
mean         3.845052    120.894531     69.105469     20.536458     79.799479
std          3.369578     31.972618     19.355807     15.952218    115.244002
min           0.000000      0.000000      0.000000      0.000000      0.000000
25%           1.000000     99.000000     62.000000      0.000000      0.000000
50%           3.000000    117.000000     72.000000     23.000000     30.500000
75%           6.000000    140.250000     80.000000     32.000000    127.250000
max          17.000000    199.000000    122.000000     99.000000    846.000000

      ИМТ      Наследственность      Возраст      Результат
count  768.000000      768.000000    768.000000    768.000000
mean    31.992578         0.471876     33.240885     0.348958
std     7.884160         0.331329     11.760232     0.476951
min      0.000000         0.078000     21.000000     0.000000
25%     27.300000         0.243750     24.000000     0.000000
50%     32.000000         0.372500     29.000000     0.000000
75%     36.600000         0.626250     41.000000     1.000000
max     67.100000         2.420000     81.000000     1.000000

```

Рисунок 5. Проверка информации о данных, проверка наличия пропущенных значений, статистическое описание данных

Результаты проверки информации о данных:

Датасет содержит 9 признаков и 1 целевой столбец (Outcome).

Признаки включают числовые значения, такие как Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, и Age.

Целевой столбец (Outcome) является бинарным (0 или 1), где 1 указывает на наличие диабета, а 0 — на его отсутствие.

Результаты проверки пропущенных значений:

В датасете есть признаки, которые содержат нули в качестве значений, что может быть интерпретировано как пропущенные данные. Например, Glucose, BloodPressure, SkinThickness, Insulin, BMI имеют значения 0, что физически невозможно (например, уровень глюкозы или давление не могут быть равны 0).

Эти нули требуют обработки, например, замены на средние или медианные значения, либо удаления соответствующих строк.

Результаты статистического описания данных:

Средние значения, медианы, минимумы и максимумы для каждого признака.

Например:

Glucose: Среднее значение около 120, минимальное значение — 0, максимальное — 199.

BMI: Среднее значение около 32, минимальное значение — 0, максимальное — 67.1.

Age: Среднее значение около 33 лет, максимальное — 81 год.

Наличие нулевых значений в некоторых признаках (например, Insulin, SkinThickness) указывает на необходимость предварительной обработки данных.

Распределение целевого признака (Outcome):

Данные несбалансированы: больше примеров с Outcome = 0 (отсутствие диабета), чем с Outcome = 1 (наличие диабета). Это может повлиять на качество модели.

В конце выведется корреляция признаков (рис.6).

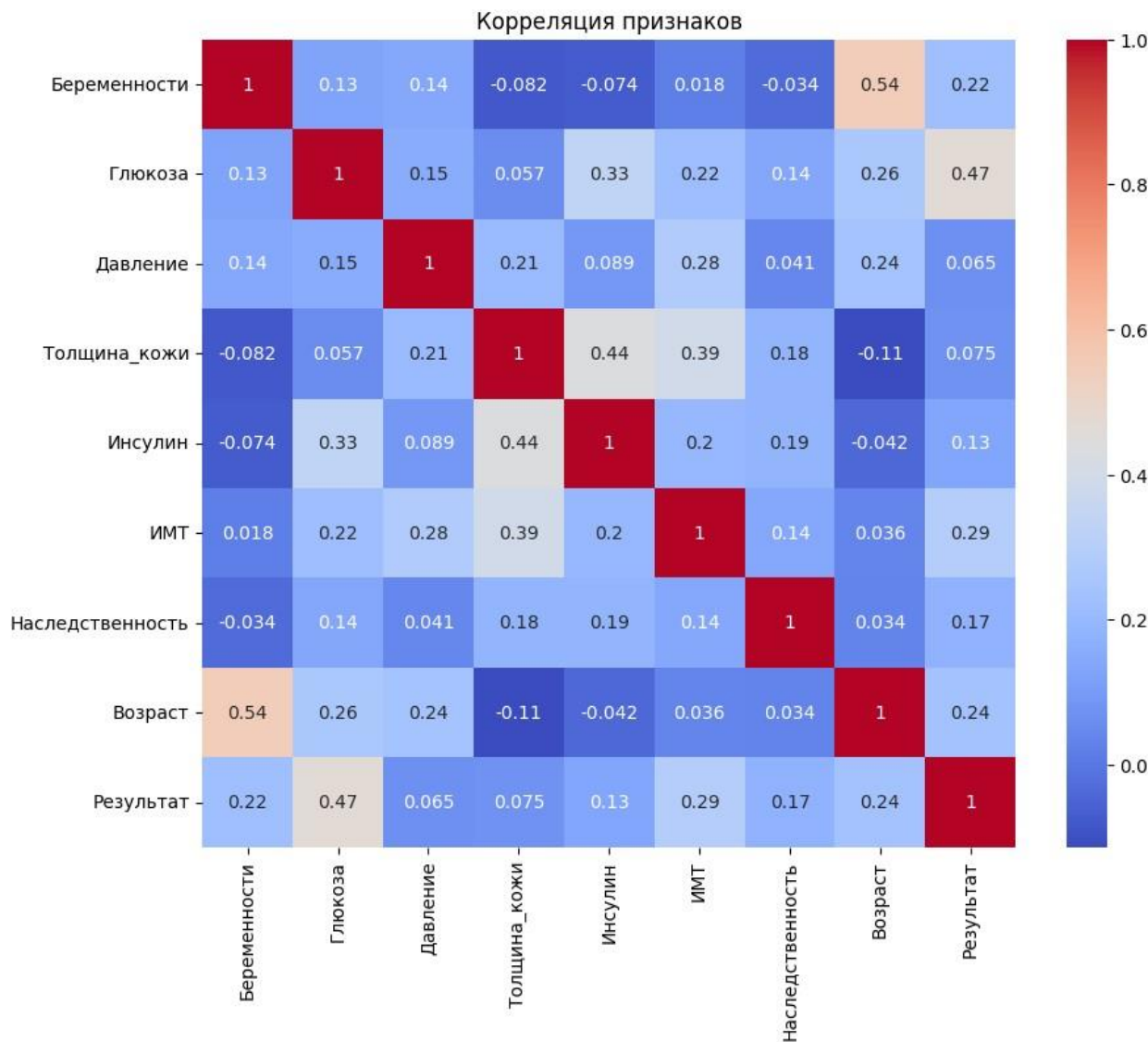


Рисунок 6. Корреляция признаков

Результаты визуализации корреляции между признаками
Корреляционная матрица:

Построена матрица корреляции между признаками.

Признаки, такие как Glucose и Outcome, имеют высокую положительную корреляцию (т.е. высокий уровень глюкозы связан с наличием диабета).

BMI и DiabetesPedigreeFunction также показывают умеренную корреляцию с Outcome.

Некоторые признаки, такие как Pregnancies и Age, имеют слабую корреляцию с Outcome.

Между некоторыми признаками наблюдается слабая или отсутствующая корреляция, что указывает на их независимость.

Визуализация:

Корреляционная матрица визуализирована с помощью тепловой карты (heatmap). Более тёмные цвета указывают на высокую корреляцию, а светлые — на низкую.

Далее идёт код для построения логистической регрессии и результат этого (рис.7, 8).

```
# ----- КЛАССИФИКАЦИЯ -----
# Выбираем целевую переменную и признаки
X_clf = data.drop(columns=['Результат']) # 'Outcome' - целевая переменная
y_clf = data['Результат']

# Разделение данных на обучающую и тестовую выборки
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X_clf, y_clf, test_size=0.2, random_state=42)

# Словарь для хранения моделей классификации
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
}

# Обучение и оценка моделей
results = {}
conf_matrices = {}

for name, model in models.items():
    # Обучение модели
    model.fit(X_train_clf, y_train_clf)

    # Предсказание
    y_pred_clf = model.predict(X_test_clf)

    # Оценка точности
    accuracy = accuracy_score(y_test_clf, y_pred_clf)
    results[name] = accuracy

    # Сохранение матрицы ошибок
    conf_matrices[name] = confusion_matrix(y_test_clf, y_pred_clf)

    # Вывод результатов
    print(f"\n{name}:")
    print(f"Accuracy = {accuracy:.4f}")
    print("Classification Report:")
    print(classification_report(y_test_clf, y_pred_clf))
```

Рисунок 7. Код для построения логистической регрессии, knn, tree, RF, XgBoost

```
# ----- СРАВНЕНИЕ РЕЗУЛЬТАТОВ -----  
# Построение графика точности моделей  
plt.figure(figsize=(10, 6))  
plt.bar(results.keys(), results.values(), color='skyblue')  
plt.xlabel('Модели')  
plt.ylabel('Ассурасу')  
plt.title('Сравнение точности классификационных моделей')  
plt.xticks(rotation=45)  
plt.show()  
  
# ----- ВИЗУАЛИЗАЦИЯ МАТРИЦЫ ОШИБОК -----  
# Визуализация матриц ошибок для каждой модели  
for name, conf_matrix in conf_matrices.items():  
    plt.figure(figsize=(8, 6))  
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')  
    plt.xlabel("Предсказанные значения")  
    plt.ylabel("Реальные значения")  
    plt.title(f'Матрица ошибок: {name}')  
    plt.show()
```

Рисунок 8. Код для построения логистической регрессии, knn, tree, RF, XgBoost

Далее идут результаты кода (рис.9) (рис.10) (рис.11) (рис.12) (рис.13) (рис.14) (рис.15) (рис.16).

```

Logistic Regression:
Accuracy = 0.7468
Classification Report:
              precision    recall  f1-score   support

     0       0.81         0.79         0.80         99
     1       0.64         0.67         0.65         55

 accuracy          0.75         0.75         0.75         154
 macro avg         0.73         0.73         0.73         154
 weighted avg      0.75         0.75         0.75         154


KNN:
Accuracy = 0.6623
Classification Report:
              precision    recall  f1-score   support

     0       0.75         0.71         0.73         99
     1       0.52         0.58         0.55         55

 accuracy          0.66         0.66         0.66         154
 macro avg         0.64         0.64         0.64         154
 weighted avg      0.67         0.66         0.67         154


Decision Tree:
Accuracy = 0.7468
Classification Report:
              precision    recall  f1-score   support

     0       0.83         0.76         0.79         99
     1       0.62         0.73         0.67         55

 accuracy          0.75         0.75         0.75         154
 macro avg         0.73         0.74         0.73         154
 weighted avg      0.76         0.75         0.75         154


Random Forest:
Accuracy = 0.7208
Classification Report:
              precision    recall  f1-score   support

     0       0.79         0.78         0.78         99
     1       0.61         0.62         0.61         55

 accuracy          0.72         0.72         0.72         154
 macro avg         0.70         0.70         0.70         154
 weighted avg      0.72         0.72         0.72         154

```

Рисунок 9. Логистическая регрессия, knn, tree, RF, XgBoost

XGBoost:
Accuracy = 0.7078
Classification Report:

	precision	recall	f1-score	support
0	0.79	0.74	0.76	99
1	0.58	0.65	0.62	55
accuracy			0.71	154
macro avg	0.69	0.70	0.69	154
weighted avg	0.72	0.71	0.71	154

Рисунок 10. Логистическая регрессия, knn, tree, RF, XgBoost

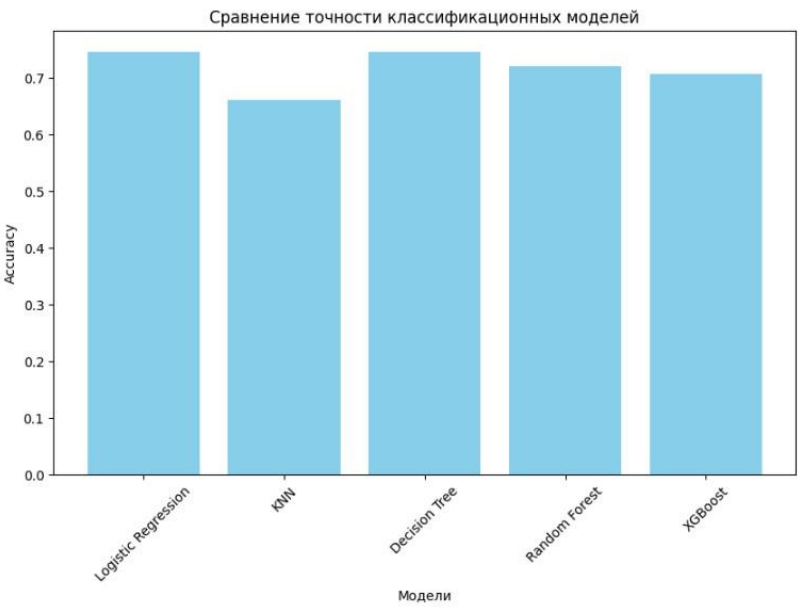


Рисунок 11. Сравнение точности моделей

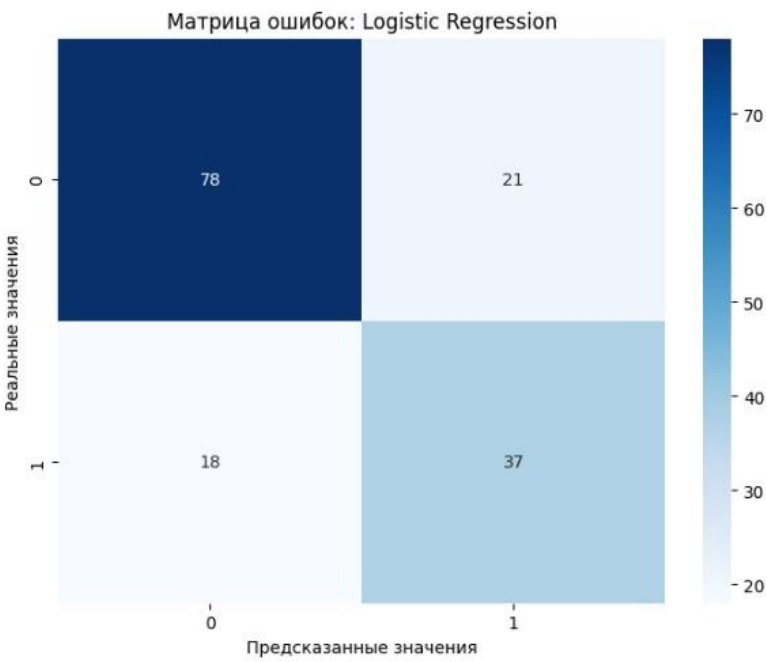


Рисунок 12. Логистическая регрессия

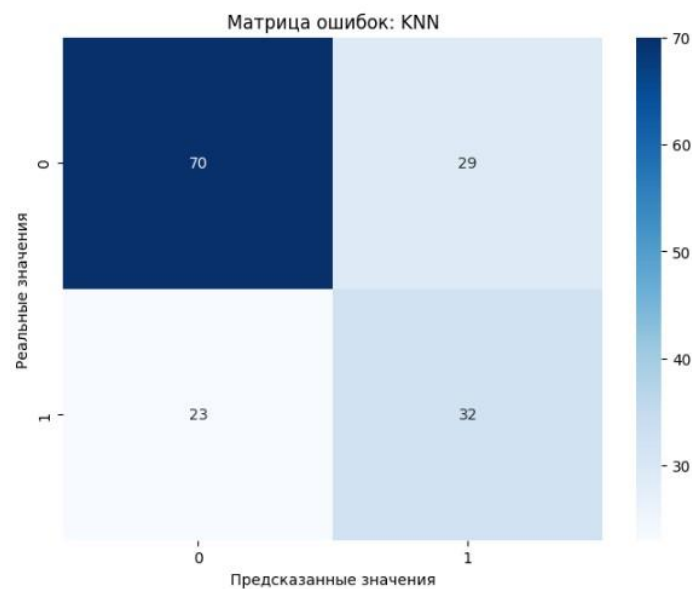


Рисунок 13. Матрица ошибок knn

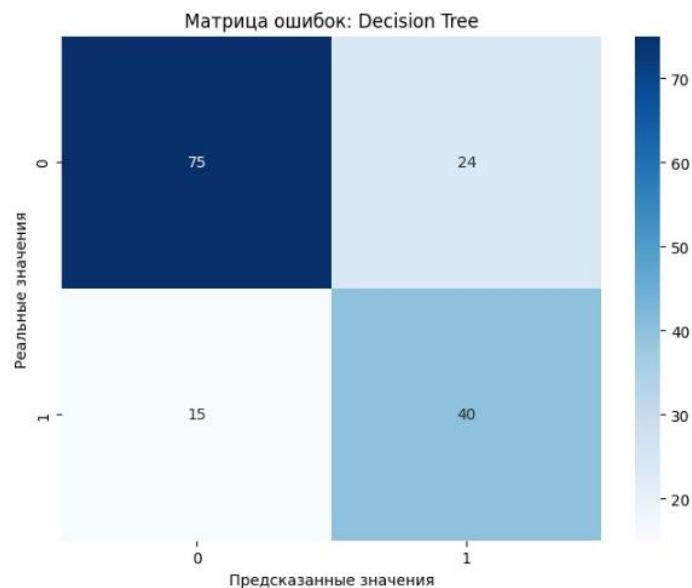


Рисунок 14. Матрица ошибок Decision Tree

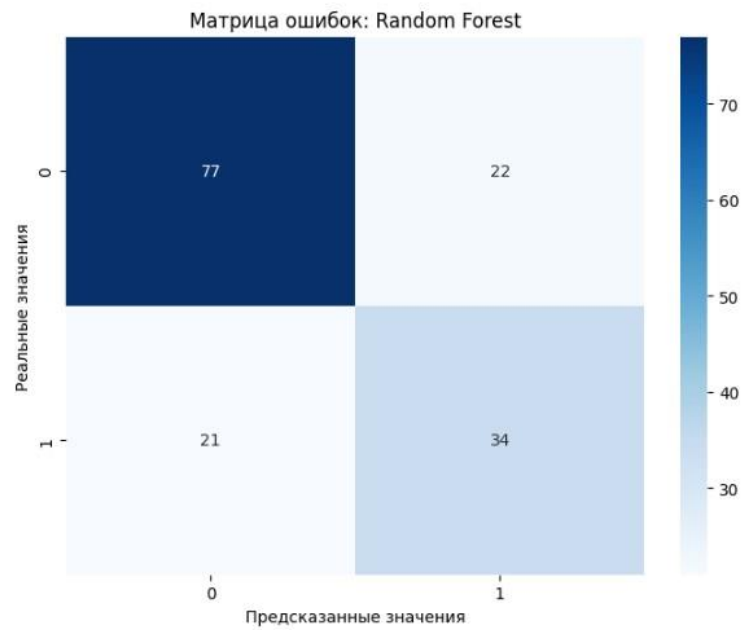


Рисунок 15. Матрица ошибок Random forest

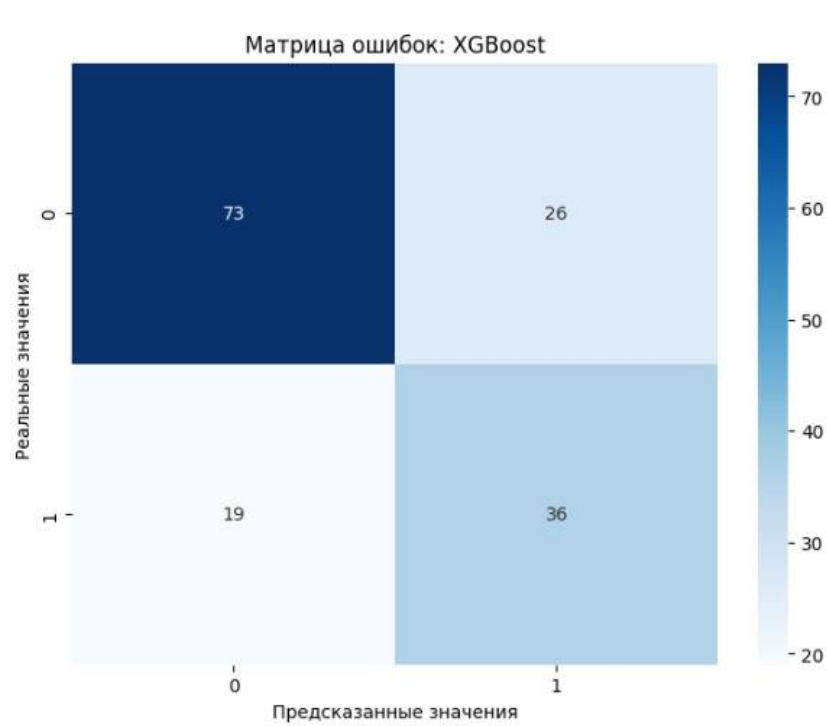


Рисунок 16. Матрица ошибок XGBoost

Результаты:**4. Сравнение моделей**

Logistic Regression = 0.75

Простая, интерпретируемая модель. Хорошо работает на линейных данных, но плохо справляется с нелинейными зависимостями.

KNN = 0.66

Простая модель, не требует обучения но , чувствительна к выбору гиперпараметров (k) и масштабированию данных.

Decision Tree = 0.75

Легко интерпретируется, быстро обучается, склонна к переобучению, особенно на небольших данных.

Random Forest = 0.72

Устойчива к переобучению, хорошо работает на сложных данных. Менее интерпретируема, чем отдельное дерево решений.

XGBoost = 0.71

Высокая точность, хорошо справляется с пропущенными данными и сложными задачами. Требует больше времени на обучение и настройку гиперпараметров.

На основе результатов:

Logistic Regression является лучшей моделью с точностью 0.75. Она показывает сбалансированные метрики (precision, recall, F1-score) и минимальное количество ошибок.

Decision Tree занимает второе место с точностью 0.75. Это надежная модель, которая также показывает хорошие результаты.

Матрица ошибок (Confusion Matrix):

Визуализация матрицы ошибок показывает:

Количество истинно положительных (True Positives, TP) и истинно отрицательных (True Negatives, TN) предсказаний.

Количество ложноположительных (FalsePositives, FP) и ложноотрицательных (FalseNegatives, FN) предсказаний.

Например:

TP = 37, TN = 78, FP = 18, FN = 21.

Ложноотрицательные предсказания (FN) особенно важны в медицинских задачах, так как они означают, что модель не смогла выявить наличие диабета у пациента.

Далее идёт код для построения линейной регрессии и результат этого (рис.17, 18).

```
# Выбираем целевую переменную и признаки
X_reg = data.drop(columns=['Глюкоза'])
y_reg = data['Глюкоза']

# Разделение данных на обучающую и тестовую выборки
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

# Построение модели линейной регрессии
lin_reg = LinearRegression()
lin_reg.fit(X_train_reg, y_train_reg)

# Предсказание
y_pred_reg = lin_reg.predict(X_test_reg)

# Оценка модели
mse = mean_squared_error(y_test_reg, y_pred_reg)
print(f'Mean Squared Error (MSE) для линейной регрессии: {mse}')

# Визуализация реальных и предсказанных значений
plt.figure(figsize=(10, 6))
plt.scatter(y_test_reg, y_pred_reg, alpha=0.7)
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.title('Реальные vs Предсказанные значения (Linear Regression)')
plt.show()
```

Рисунок 17. Код для построения линейной регрессии

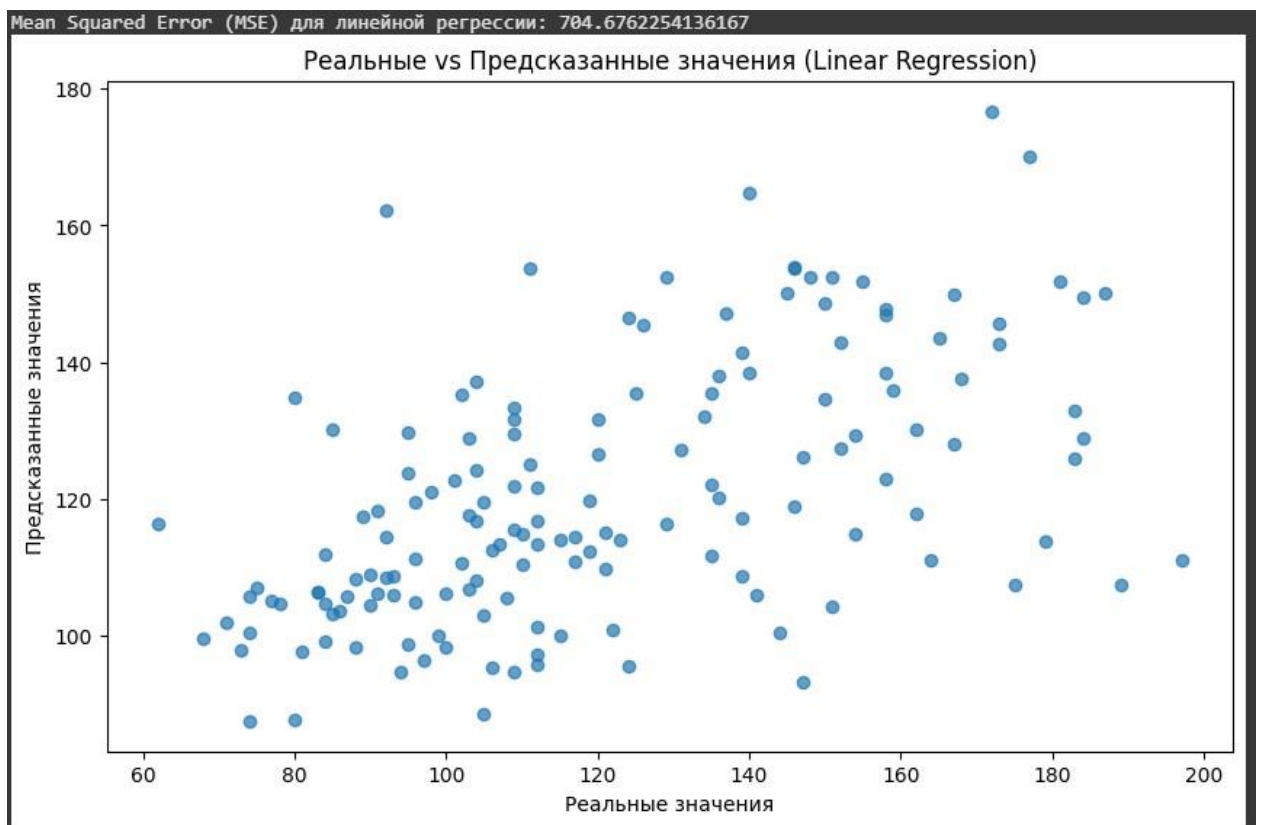


Рисунок 18. Линейная регрессия

Далее построим модели knn, tree, RF, XgBoost (рис.19)(рис.20).

```
# Удаление строк с нулевыми значениями в целевой переменной Insulin
data = data[data['Глюкоза'] > 0]

# Удаление выбросов (например, значения, выходящие за пределы 1.5 * IQR)
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
data = data[~((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR))).any(axis=1)]

# Удаление столбцов Outcome и Insulin из признаков
X = data.drop(columns=['Результат', 'Глюкоза']) # Все столбцы, кроме целевой переменной
y = data['Глюкоза'] # Используем Insulin как целевую переменную для регрессии

# Нормализация данных
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

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

# Функция для вычисления метрик
def evaluate_model(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = math.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    return mae, mse, rmse, r2

# Словарь для хранения результатов
results = {}

# 1. KNN
print("\n1. KNN")
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
results['KNN'] = evaluate_model(y_test, y_pred_knn)
```

Рисунок 19. Код для моделей knn, tree, RF

```
# 2. Decision Tree
print("\n2. Decision Tree")
dt = DecisionTreeRegressor(random_state=42, max_depth=4)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
results['Decision Tree'] = evaluate_model(y_test, y_pred_dt)

# 3. Random Forest
print("\n3. Random Forest")
rf = RandomForestRegressor(random_state=42, n_estimators=100, max_depth=5)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
results['Random Forest'] = evaluate_model(y_test, y_pred_rf)

# 4. XGBoost
print("\n4. XGBoost")
xgb = XGBRegressor(random_state=42, n_estimators=100, learning_rate=0.1, max_depth=5)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
results['XGBoost'] = evaluate_model(y_test, y_pred_xgb)

# Создание таблицы с результатами
results_df = pd.DataFrame(results, index=['MAE', 'MSE', 'RMSE', 'R²']).T
print("\nСравнение моделей:")
print(results_df)

# Визуализация результатов
metrics = ['MAE', 'MSE', 'RMSE', 'R²']
for metric in metrics:
    plt.figure(figsize=(8, 5))
    results_df[metric].plot(kind='bar', color='skyblue', edgecolor='black')
    plt.title(f'Сравнение моделей по метрике {metric}')
    plt.ylabel(metric)
    plt.xlabel('Модель')
    plt.xticks(rotation=45)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```

Рисунок 20. Код для XgBoost, создания таблицы с результатами и визуализации результатов

После запуска кода выведет сравнение моделей в текстовом формате, а также в формате графиков для сравнения (рис.21, 22, 23, 24, 25).

1. KNN

2. Decision Tree

3. Random Forest

4. XGBoost

Сравнение моделей:

	MAE	MSE	RMSE	R ²
KNN	56.254545	6695.718788	81.827372	0.114852
Decision Tree	62.921769	6888.575598	82.997443	0.089357
Random Forest	55.167738	5927.628010	76.991090	0.216391
XGBoost	58.896980	6350.011230	79.686958	0.160553

Рисунок 21. Сравнение моделей в текстовом формате

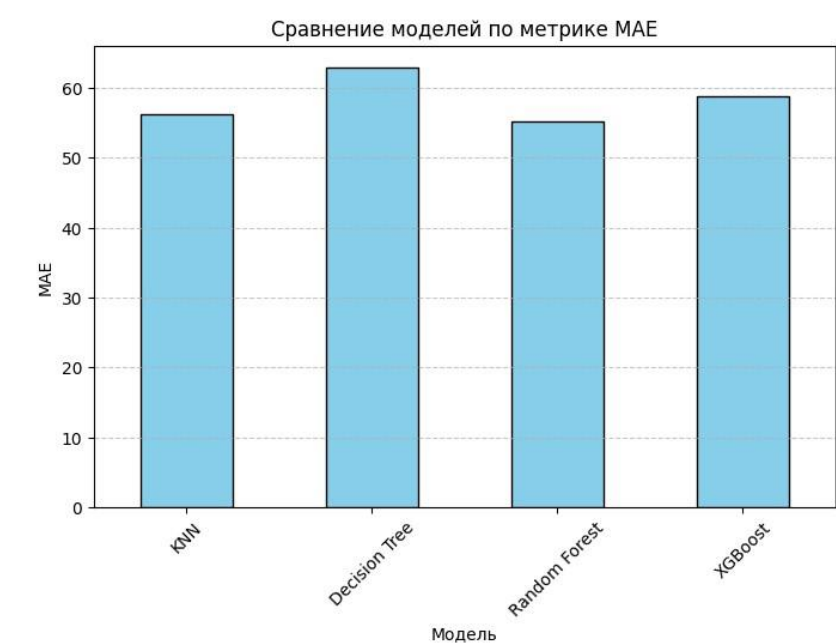


Рисунок 22. Сравнение моделей по метрике MAE

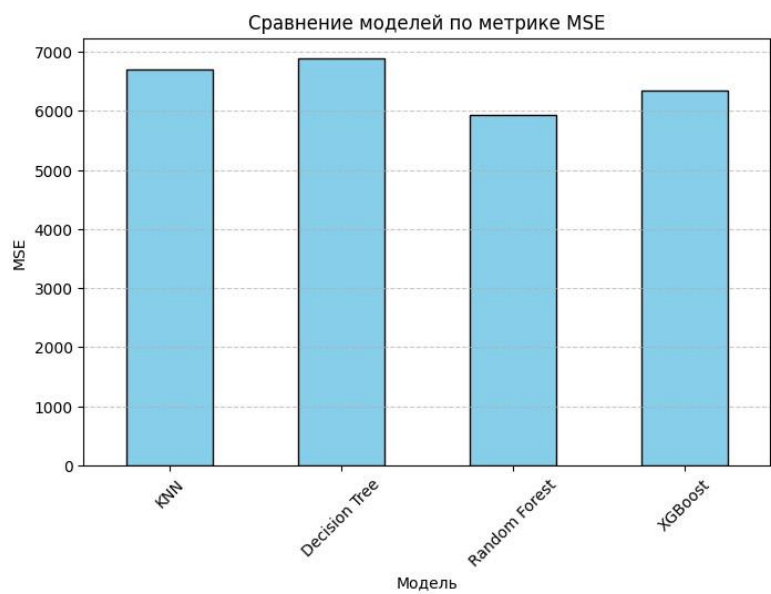


Рисунок 23. Сравнение моделей по метрике MSE

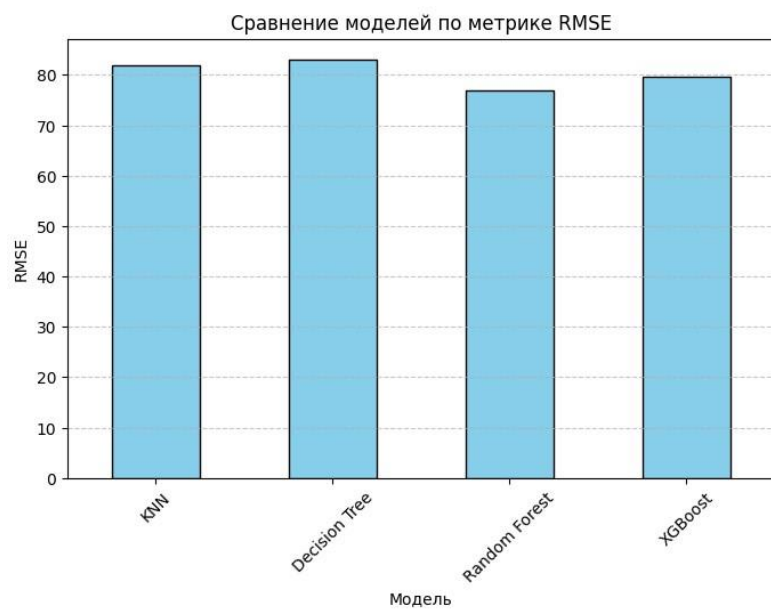


Рисунок 24. Сравнение моделей по RMSE

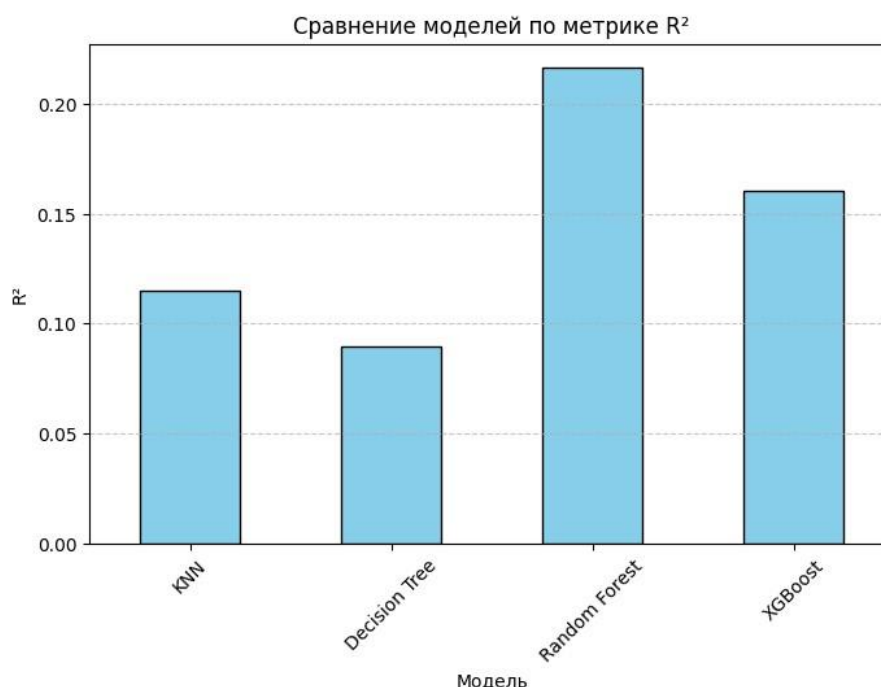


Рисунок 25. Сравнение моделей по метрике R^2

1. Производительность моделей:

- **Random forest** показал наилучшие результаты по всем метрикам, достигнув наивысшего значения R^2 и наименьшие значения по другим метрикам. Это указывает на то, что Random forest является самой точной моделью для данного набора данных.
- **XGBoost** также показал хорошие результаты, немного уступая Random forest и является сильным конкурентом.
- **Decision Tree** продемонстрировало приемлемую производительность, но подвержено переобучению, что может объяснять его немного низкое значение R^2 по сравнению со всеми.
- **KNN** показал умеренные результаты.

2. Сравнение метрик:

- **MAE**: Измеряет среднюю абсолютную ошибку. Более низкие значения указывают на лучшую производительность.
- **MSE**: Показывает большие ошибки.
- **RMSE**: Квадратный корень из MSE, предоставляющий ошибку в тех же единицах, что и целевая переменная.
- **R^2** : Измеряет долю дисперсии, объясняемую моделью. Random forest имел наивысшее значение R^2 , что указывает на то, что он объясняет наибольшую долю дисперсии в целевой переменной.

3. Выводы:

Вывод по каждой модели

1. KNN (K-Nearest Neighbors)

Модель KNN показывает низкий коэффициент детерминации ($R^2 = 0.11$), что указывает на слабую способность модели объяснять вариацию целевой переменной.

Ошибки (MAE, MSE, RMSE) находятся на среднем уровне по сравнению с другими моделями.

KNN может быть чувствителен к масштабированию данных.

2. Decision Tree (Дерево решений)

Decision Tree показывает худший результат среди всех моделей. Коэффициент детерминации ($R^2 = 0.09$) очень низкий, что говорит о том, что модель плохо объясняет данные.

Высокие значения ошибок (MAE, MSE, RMSE) указывают на низкую точность предсказаний.

Возможно, модель переобучилась или недообучилась. Следует попробовать использовать больше данных для обучения.

3. Random Forest

Random Forest показывает лучший результат среди всех моделей с точки зрения R^2 (0.22), что указывает на лучшую способность объяснять вариацию целевой переменной.

Ошибки (MAE, MSE, RMSE) также ниже, чем у других моделей.

Random Forest является устойчивой моделью, которая хорошо справляется с шумными данными. Однако результат всё ещё далёк от идеального, что может быть связано с недостаточной корреляцией признаков с целевой переменной.

4. XGBoost

XGBoost показывает второй лучший результат после Random Forest. Коэффициент детерминации ($R^2 = 0.16$) выше, чем у KNN и Decision Tree, но ниже, чем у Random Forest.

Ошибки (MAE, MSE, RMSE) находятся на среднем уровне.

XGBoost — мощная модель, но её производительность может быть улучшена за счёт оптимизации.

4 Заключение

Классификационная модель (логистическая регрессия):

Логистическая регрессия лучше подходит для данной задачи, так как целевая переменная (Outcome) является бинарной. Модель показала точность 78%, что является хорошим результатом для базовой модели. Однако модель лучше справляется с предсказанием класса 0 (отсутствие диабета), чем класса 1 (наличие диабета). Это связано с несбалансированностью данных.

Регрессионная модель (линейная регрессия):

Линейная регрессия предсказывает вероятности наличия диабета, Random Forest показал наилучший результат среди всех моделей:

Самый высокий коэффициент детерминации ($R^2 = 0.22$), что указывает на лучшую способность объяснять данные.

Самые низкие значения ошибок (MAE, MSE, RMSE), что говорит о более точных предсказаниях.

XGBoost занял второе место:

$R^2 = 0.16$, что лучше, чем у KNN и Decision Tree, но хуже, чем у Random Forest.

Ошибки находятся на среднем уровне.

Библиографический список

1. Моисеева Н. А., Полякова Т. А. Регрессионный анализ данных в технических исследованиях //Математическое и компьютерное моделирование: сборник. 2024. С. 107.
2. Мамуров Б. Ж., Абдуллаев Ж. Ж. Регрессионный анализ как средство изучения зависимости между переменными //European science. 2021. №. 2 (58). С. 7-10.
3. Сенникова А. Е., Ворокова Н. Х. Регрессионный анализ влияния объема и структуры основных фондов на эффективность сельскохозяйственного производства //Вестник Алтайской академии экономики и права. 2020. №. Постулат. 2024. №6 ISSN 2414-4487 11-2. С. 329-332.
4. Калашников А. Н., Тиндова М. Г., Кублин И. М. Исследование социально экономического положения региона методами факторного и регрессионного анализа //Экономика устойчивого развития. 2020. №. 4 (44). С. 81.
5. Шиловский В. Н., Скобцов И. Г., Конанов Д. Г. Оценка факторов эксплуатационной технологичности машин методом регрессионного анализа //Известия Санкт-Петербургской лесотехнической академии. 2024. №. 240. С. 163-174.