Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

Выполнил:
Каятский П. Е.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 13.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие ансамблевые модели:
    - две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
    - AdaBoost;
    - градиентный бустинг.
5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Ссылка на датасет, используемый в лр:

https://www.kaggle.com/datasets/khushikyad001/water-pollution-and-disease

Ход выполнения:

Лабораторная работа №5 ч.1

Анализ ансамблевых методов машинного обучения

1. Загрузка и подготовка данных

[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, LabelEncoder
from sklearn.ensemble import (RandomForestRegressor, GradientBoostingRegressor,
                              AdaBoostRegressor, BaggingRegressor, ExtraTreesRegressor)
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer

```
Executed at 2025.04.13 20:13:14 in 3s 95ms

[3]
```
# Загрузка данных
data = pd.read_csv('water_pollution_disease.csv')

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

```
Executed at 2025.04.13 20:14:36 in 51ms

```
       Country   Region  Year Water Source Type  Contaminant Level (ppm)  \
0       Mexico    North  2015              Lake                     6.06
1       Brazil     West  2017              Well                     5.24
2    Indonesia  Central  2022              Pond                     0.24
3      Nigeria     East  2016              Well                     7.91
4       Mexico    South  2005              Well                     0.12

   pH Level  Turbidity (NTU)  Dissolved Oxygen (mg/L)  Nitrate Level (mg/L)  \
0      7.12             3.93                     4.28                  8.28
1      7.84             4.79                     3.86                 15.74
2      6.43             0.79                     3.42                 36.67
```

[4]
```
# Основная информация о данных
print("\nИнформация о данных:")
print(data.info())
```
Executed at 2025.04.13 20:14:50 in 27ms

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 24 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Country                                 3000 non-null   object
 1   Region                                  3000 non-null   object
 2   Year                                    3000 non-null   int64
 3   Water Source Type                       3000 non-null   object
```

[5]
```
# Проверка пропущенных значений
print("\nПропущенные значения:")
print(data.isnull().sum())
```

Executed at 2025.04.13 20:15:02 in 13ms

```
Пропущенные значения:
Country                                0
Region                                 0
Year                                   0
Water Source Type                      0
Contaminant Level (ppm)                0
pH Level                               0
Turbidity (NTU)                        0
Dissolved Oxygen (mg/L)                0
Nitrate Level (mg/L)                   0
```

[6]
```
# Описательная статистика
print("\nОписательная статистика:")
print(data.describe())
```

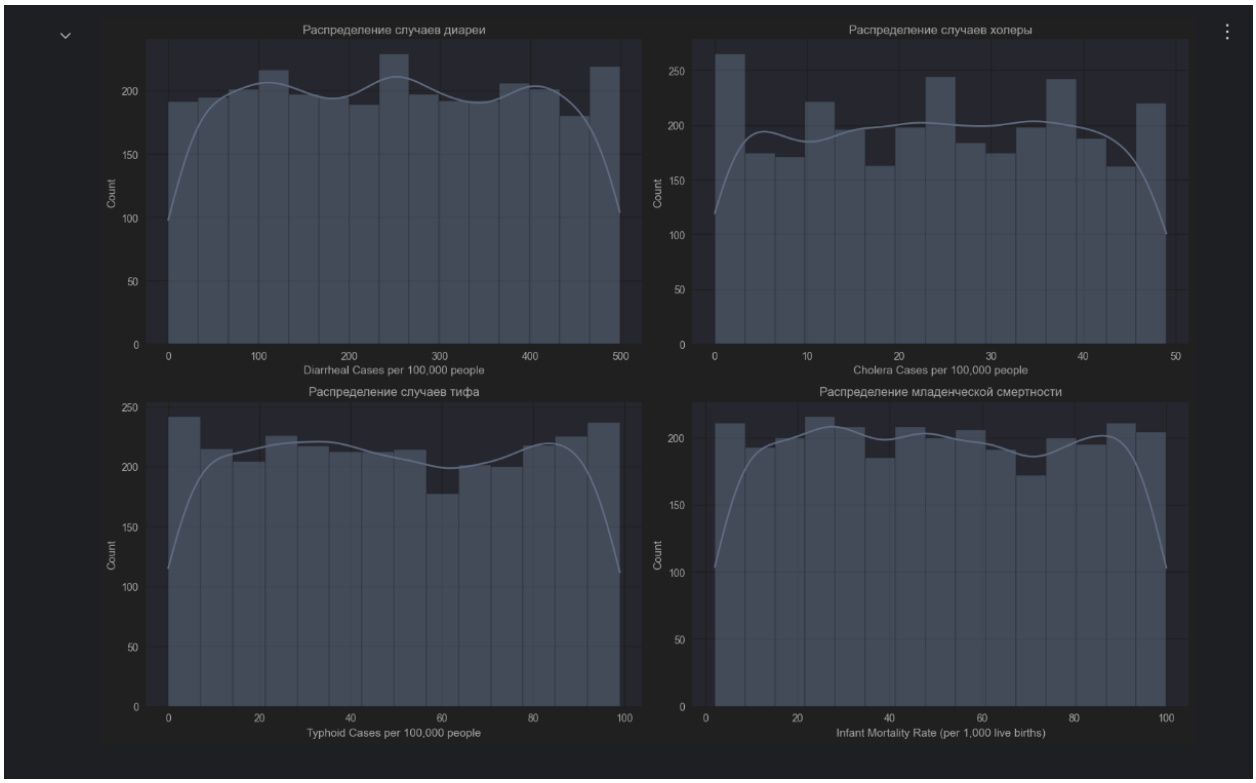
Executed at 2025.04.13 20:15:08 in 420ms

```
              Year  Contaminant Level (ppm)  pH Level  Turbidity (NTU)
count  3000.000000              3000.000000  3000.000000      3000.000000
mean   2012.012667                 4.954390     7.255847         2.480023
std       7.229287                 2.860072     0.720464         1.419984
min    2000.000000                 0.000000     6.000000         0.000000
25%    2006.000000                 2.560000     6.630000         1.257500
50%    2012.000000                 4.950000     7.280000         2.460000
75%    2018.000000                 7.400000     7.870000         3.660000
max    2024.000000                10.000000     8.500000         4.990000

       Dissolved Oxygen (mg/L)  Nitrate Level (mg/L)  \
count              3000.000000           3000.00000
```

2. Визуализация данных

```
[8]
# Настройка стиля графиков
plt.style.use('seaborn-v0_8')  # Или другой доступный стиль из plt.style.available
plt.figure(figsize=(15, 10))

# Распределение случаев заболеваний
plt.subplot(2, 2, 1)
sns.histplot(data['Diarrheal Cases per 100,000 people'], kde=True)
plt.title('Распределение случаев диареи')

plt.subplot(2, 2, 2)
sns.histplot(data['Cholera Cases per 100,000 people'], kde=True)
plt.title('Распределение случаев холеры')

plt.subplot(2, 2, 3)
sns.histplot(data['Typhoid Cases per 100,000 people'], kde=True)
plt.title('Распределение случаев тифа')

plt.subplot(2, 2, 4)
sns.histplot(data['Infant Mortality Rate (per 1,000 live births)'], kde=True)
plt.title('Распределение младенческой смертности')

plt.tight_layout()
plt.show()
```

Executed at 2025.04.13 20:17:15 in 2s 261ms



Корреляция между параметрами воды и заболеваниями

```
[9]
water_params = ['Contaminant Level (ppm)', 'pH Level', 'Turbidity (NTU)',
                'Dissolved Oxygen (mg/L)', 'Nitrate Level (mg/L)',
                'Lead Concentration (µg/L)', 'Bacteria Count (CFU/mL)']

diseases = ['Diarrheal Cases per 100,000 people',
            'Cholera Cases per 100,000 people',
            'Typhoid Cases per 100,000 people',
            'Infant Mortality Rate (per 1,000 live births)']

plt.figure(figsize=(12, 8))
corr_matrix = data[water_params + diseases].corr()
sns.heatmap(corr_matrix[diseases].loc[water_params], annot=True, cmap='coolwarm', center=0)
plt.title('Корреляция параметров воды с заболеваниями')
plt.show()
```

Executed at 2025.04.13 20:18:06 in 505ms

3. Подготовка данных для моделирования

Add Code Cell | Add Markdown Cell

Обработка пропущенных значений

```
[10]
imputer = SimpleImputer(strategy='median')
data_imputed = pd.DataFrame(imputer.fit_transform(data.select_dtypes(include=[np.number])),
                            columns=data.select_dtypes(include=[np.number]).columns)
```

Executed at 2025.04.13 20:18:19 in 24ms

Кодирование категориальных переменных

```
[11]
cat_cols = data.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_cols:
    data_imputed[col] = le.fit_transform(data[col])
```

Executed at 2025.04.13 20:18:20 in 15ms

Выбор признаков и целевых переменных

```
[12]
features = data_imputed.drop(diseases + ['Country', 'Region', 'Year'], axis=1)
target_diarrhea = data_imputed['Diarrheal Cases per 100,000 people']
target_cholera = data_imputed['Cholera Cases per 100,000 people']
target_typhoid = data_imputed['Typhoid Cases per 100,000 people']
target_infant = data_imputed['Infant Mortality Rate (per 1,000 live births)']
# Масштабирование признаков
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train_d, y_test_d = train_test_split(features_scaled, target_diarrhea, test_size=0.3, random_state=42)
_, _, y_train_c, y_test_c = train_test_split(features_scaled, target_cholera, test_size=0.3, random_state=42)
_, _, y_train_t, y_test_t = train_test_split(features_scaled, target_typhoid, test_size=0.3, random_state=42)
_, _, y_train_i, y_test_i = train_test_split(features_scaled, target_infant, test_size=0.3, random_state=42)
```

Executed at 2025.04.13 20:18:23 in 32ms

4. Обучение моделей для прогнозирования диареи

```
[14]
# Инициализация моделей
models = {
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100, random_state=42),
    'AdaBoost': AdaBoostRegressor(n_estimators=100, random_state=42),
    'Bagging': BaggingRegressor(n_estimators=100, random_state=42),
    'Extra Trees': ExtraTreesRegressor(n_estimators=100, random_state=42)
}

# Обучение и оценка моделей для диареи
results_list = []

for name, model in models.items():
    model.fit(X_train, y_train_d)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test_d, y_pred))
    r2 = r2_score(y_test_d, y_pred)
    results_list.append({
        'Model': name,
        'RMSE': rmse,
        'R2': r2
    })

# Создание DataFrame из списка словарей
results_diarrhea = pd.DataFrame(results_list)
```

```
# Создание DataFrame из списка словарей
results_diarrhea = pd.DataFrame(results_list)

print("Результаты для случаев диареи:")
print(results_diarrhea.sort_values('R2', ascending=False))
```

Executed at 2025.04.13 20:20:59 in 14s 520ms

```
Результаты для случаев диареи:
                Model        RMSE        R2
2           AdaBoost  146.606729 -0.002550
0      Random Forest  148.004502 -0.021759
3            Bagging  148.121969 -0.023381
1  Gradient Boosting  148.482596 -0.028370
4        Extra Trees  148.497955 -0.028583
```

5. Обучение моделей для прогнозирования холеры

```
[18]
results_list = []  # Создаем пустой список для хранения результатов

for name, model in models.items():
    model.fit(X_train, y_train_c)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test_c, y_pred))
    r2 = r2_score(y_test_c, y_pred)

    # Добавляем словарь с результатами в список
    results_list.append({
        'Model': name,
        'RMSE': rmse,
        'R2': r2
    })

# Создаем DataFrame из списка результатов
results_cholera = pd.DataFrame(results_list)

print("\nРезультаты для случаев холеры:")
print(results_cholera.sort_values('R2', ascending=False))
```

Executed at 2025.04.13 20:23:15 in 14s 31ms

```
Результаты для случаев холеры:
              Model       RMSE        R2
2          AdaBoost  14.368502  0.000976
1  Gradient Boosting  14.437435 -0.008633
4        Extra Trees  14.566341 -0.026725
3            Bagging  14.570095 -0.027254
0      Random Forest  14.599474 -0.031401
```

6. Обучение моделей для прогнозирования тифа

```
[19]
# Обучение и оценка моделей для тифа
results_list = []  # Создаем пустой список для хранения результатов

for name, model in models.items():
    model.fit(X_train, y_train_t)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test_t, y_pred))
    r2 = r2_score(y_test_t, y_pred)

    # Добавляем словарь с результатами в список
    results_list.append({
        'Model': name,
        'RMSE': rmse,
        'R2': r2
    })

# Создаем DataFrame из списка результатов
results_typhoid = pd.DataFrame(results_list)

print("\nРезультаты для случаев тифа:")
print(results_typhoid.sort_values('R2', ascending=False))
```

Executed at 2025.04.13 20:23:29 in 14s 2ms

```
Результаты для случаев тифа:
              Model       RMSE        R2
2          AdaBoost  29.103431 -0.002532
0      Random Forest  29.443348 -0.026087
3            Bagging  29.461885 -0.027379
1  Gradient Boosting  29.571551 -0.035042
4        Extra Trees  29.584358 -0.035938
```

7. Обучение моделей для прогнозирования младенческой смертности

[21]
```python
# Обучение и оценка моделей для младенческой смертности
results_list = []  # Создаем пустой список для хранения результатов

for name, model in models.items():
    model.fit(X_train, y_train_i)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test_i, y_pred))
    r2 = r2_score(y_test_i, y_pred)

    results_list.append({
        'Model': name,
        'RMSE': rmse,
        'R2': r2
    })

results_infant = pd.DataFrame(results_list)

# Выводим результаты, отсортированные по R2
print("\nРезультаты для младенческой смертности:")
print(results_infant.sort_values('R2', ascending=False))
```

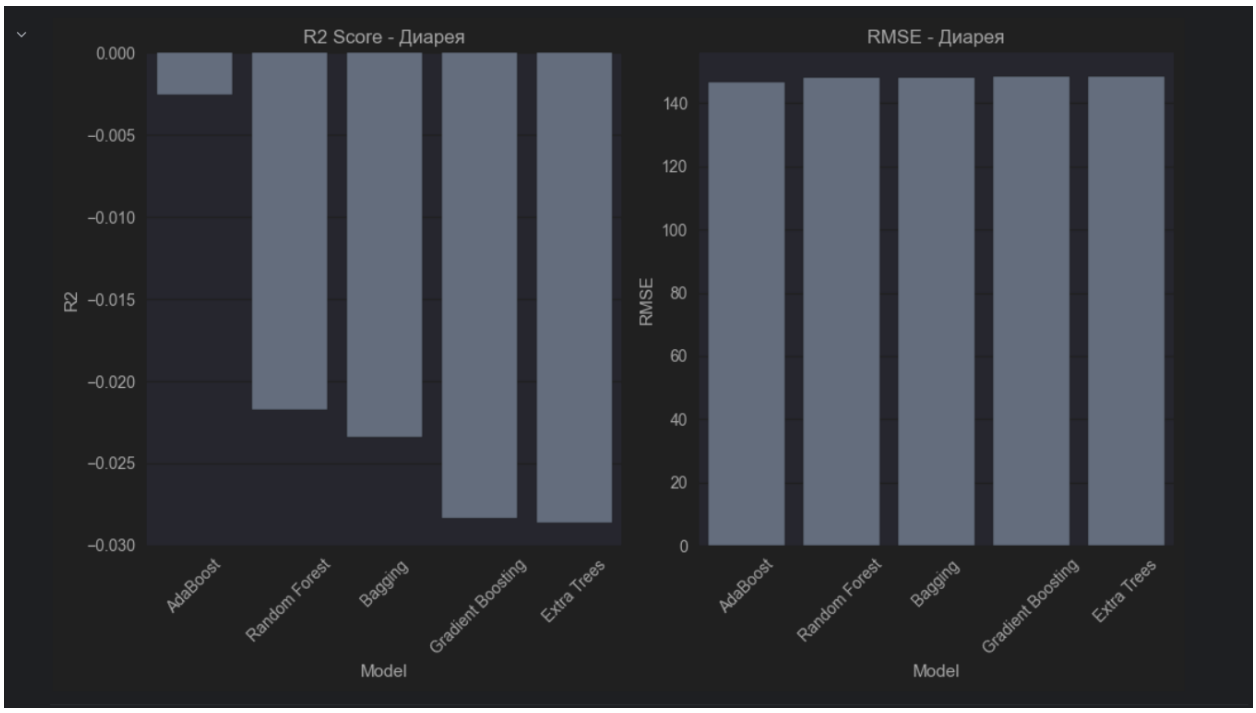
Executed at 2025.04.13 20:24:55 in 14s 223ms

```
Результаты для младенческой смертности:
               Model       RMSE        R2
2           AdaBoost  28.339900 -0.005070
1  Gradient Boosting  28.688550 -0.029952
0      Random Forest  28.728307 -0.032809
3            Bagging  28.751112 -0.034449
4        Extra Trees  28.788601 -0.037149
```

8. Визуализация результатов

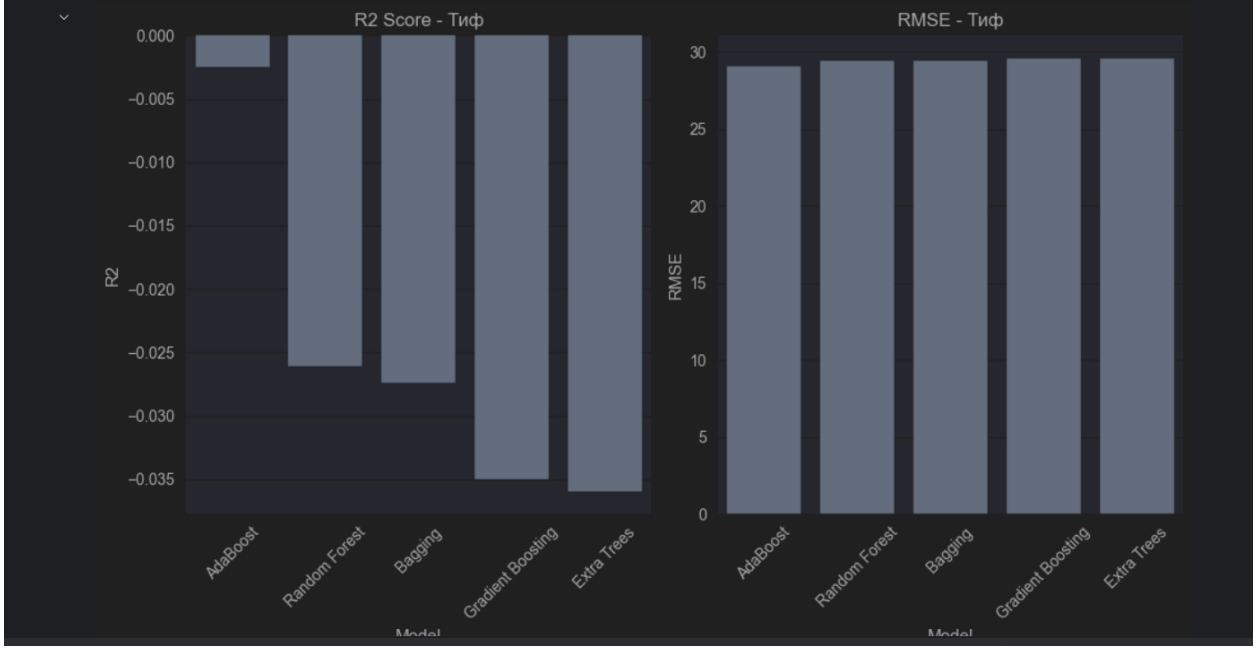
[23]
```python
# Функция для визуализации результатов
def plot_results(results, title):
    plt.figure(figsize=(10, 6))
    plt.subplot(1, 2, 1)
    sns.barplot(x='Model', y='R2', data=results.sort_values('R2', ascending=False))
    plt.title(f'R2 Score - {title}')
    plt.xticks(rotation=45)

    plt.subplot(1, 2, 2)
    sns.barplot(x='Model', y='RMSE', data=results.sort_values('RMSE'))
    plt.title(f'RMSE - {title}')
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()

plot_results(results_diarrhea, 'Диарея')
```

Executed at 2025.04.13 20:25:48 in 656ms

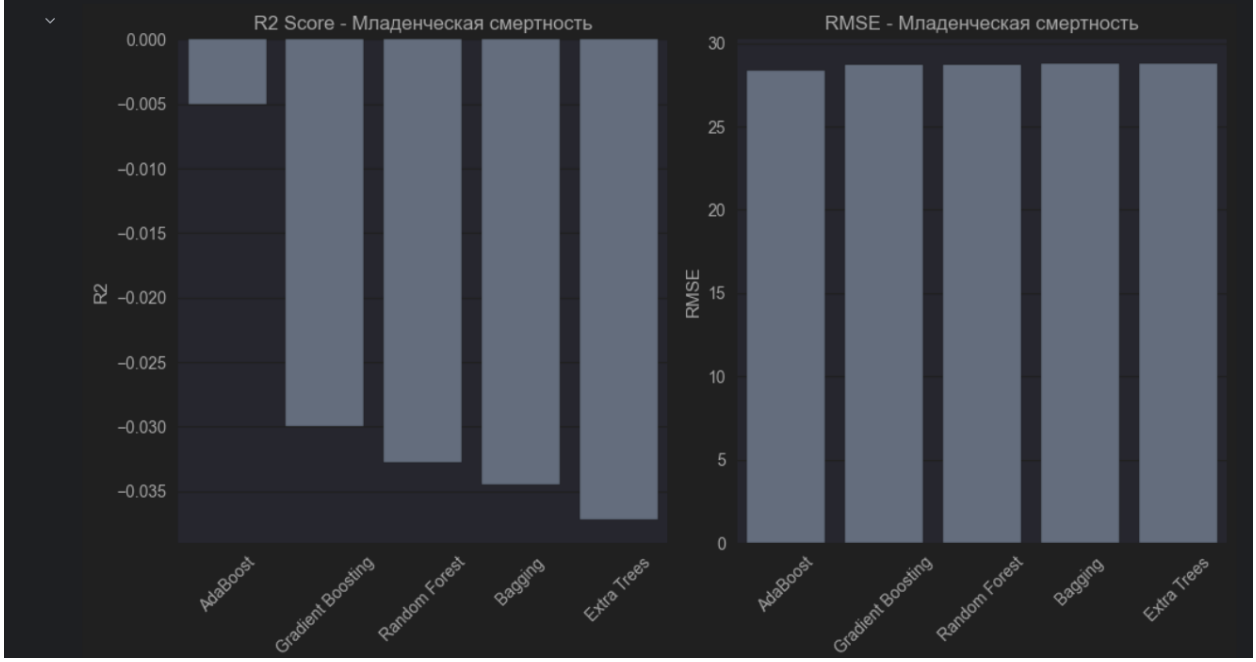

R2 Score - Диарея
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
R2
AdaBoost
Random Forest
Bagging
Gradient Boosting
Extra Trees
Model
RMSE - Диарея
140
120
100
80
60
40
20
0
RMSE
AdaBoost
Random Forest
Bagging
Gradient Boosting
Extra Trees
Model


8
190
[24]
plot_results(results_cholera, 'Холера')
Executed at 2025.04.13 20:25:53 in 629ms
R2 Score - Холера
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
R2
AdaBoost
Gradient Boosting
Extra Trees
Bagging
Random Forest
Model
RMSE - Холера
14
12
10
8
6
4
2
0
RMSE
AdaBoost
Gradient Boosting
Extra Trees
Bagging
Random Forest
Model

```
[25] 1 plot_results(results_typhoid, 'Тиф')
```
Executed at 2025.04.13 20:25:56 in 1s 117ms




```
[26] 1 plot_results(results_infant, 'Младенческая смертность')
```
Executed at 2025.04.13 20:25:58 in 783ms



9. Анализ важности признаков

```
[27]
# Анализ важности признаков для лучшей модели (Random Forest)
best_model = RandomForestRegressor(n_estimators=100, random_state=42)
best_model.fit(X_train, y_train_d)

importances = best_model.feature_importances_
feature_importance = pd.DataFrame({'Feature': features.columns, 'Importance': importances})
feature_importance = feature_importance.sort_values('Importance', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance)
plt.title('Важность признаков для прогнозирования случаев диареи')
plt.show()
```

Executed at 2025.04.13 20:26:08 in 5s 977ms