Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения.»

Вариант № 5

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

Проверил:
Гапанюк Ю.Е.

Дата: 13.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Задача №1.

ля заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Набор данных:

https://www.kaggle.com/datasets/khushikyad001/world-happiness-report

Дополнительные требования:

ИУ5-64Б, ИУ5Ц-84Б Линейная/логистическая регрессия Градиентный бустинг

Ход работы:

Рубежный контроль №2

Линейная/логистическая регрессия и Градиентный бустинг для world_hapiness_resort.csv

1. Загрузка и предварительный анализ 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
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

Executed at 2025.04.13 20:44:59 in 4s 95ms

Загрузка и анализ данных

```
[3]
data = pd.read_csv('world_happiness_report.csv')
print(f"Размер датасета: {data.shape}")
```

Executed at 2025.04.13 20:45:54 in 39ms

```
Размер датасета: (4000, 24)
```

```
[4]
print("\nПервые 5 строк:")
print(data.head())
```

Executed at 2025.04.13 20:45:56 in 35ms

```
Первые 5 строк:
  Country  Year  Happiness_Score  GDP_per_Capita  Social_Support  \
0   China  2022             4.39        44984.68            0.53
1      UK  2015             5.49        30814.59            0.93
2  Brazil  2009             4.65        39214.84            0.03
3  France  2019             5.20        30655.75            0.77
4   China  2022             7.28        30016.87            0.05

   Healthy_Life_Expectancy  Freedom  Generosity  Corruption_Perception  \
0                    71.11     0.41       -0.05                   0.83
```

```
3              57.65                    4.43          90.59
```

```
[5]  1  print("\nИнформация о данных:")
     2  print(data.info())
```
Executed at 2025.04.13 20:46:00 in 28ms

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 24 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Country                   4000 non-null   object
 1   Year                      4000 non-null   int64
 2   Happiness_Score           4000 non-null   float64
 3   GDP_per_Capita            4000 non-null   float64
```

```
[6]  1  print("\nОписательная статистика:")
     2  print(data.describe())
```
Executed at 2025.04.13 20:46:03 in 144ms

```
Описательная статистика:
              Year  Happiness_Score  GDP_per_Capita  Social_Support  \
count  4000.000000      4000.000000     4000.000000     4000.000000
mean   2014.670750         5.455005    30482.009953        0.505860
std       5.724075         1.427370    17216.122032        0.286202
min    2005.000000         3.000000     1009.310000        0.000000
25%    2010.000000         4.237500    15425.125000        0.260000
50%    2015.000000         5.430000    29991.255000        0.510000
75%    2020.000000         6.662500    45763.085000        0.750000
max    2024.000000         8.000000    59980.720000        1.000000
```

Проверка пропущенных значений

```
[7]  1  print("Пропущенные значения:")
     2  print(data.isnull().sum())
```
Executed at 2025.04.13 20:46:07 in 13ms

```
Пропущенные значения:
Country                     0
Year                        0
Happiness_Score             0
GDP_per_Capita              0
Social_Support              0
Healthy_Life_Expectancy     0
Freedom                     0
Generosity                  0
Corruption_Perception       0
Unemployment_Rate           0
```

Как мы видим, пропущенных значений в датасете не обнаружено

```
[8]
numeric_cols = data.select_dtypes(include=[np.number]).columns
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].median())

# Кодирование категориальной переменной Country
data = pd.get_dummies(data, columns=['Country'], drop_first=True)

# Выделение признаков и целевой переменной
X = data.drop('Happiness_Score', axis=1)
y = data['Happiness_Score']

# Масштабирование признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

Executed at 2025.04.13 20:46:08 in 82ms

3. Построение и оценка моделей

Линейная регрессия

```
[9]
# Создание и обучение модели
lr = LinearRegression()
lr.fit(X_train, y_train)

# Прогнозирование
y_pred_lr = lr.predict(X_test)

# Оценка качества
mse_lr = mean_squared_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mse_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

print("\nЛинейная регрессия:")
print(f"RMSE: {rmse_lr:.4f}")
print(f"MAE: {mae_lr:.4f}")
print(f"R²: {r2_lr:.4f}")
```

Executed at 2025.04.13 20:46:13 in 391ms

```
Линейная регрессия:
RMSE: 1.4437
MAE: 1.2501
R²: -0.0128
```

Градиентный бустинг

[10]
```
# Создание и обучение модели
gb = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb.fit(X_train, y_train)

# Прогнозирование
y_pred_gb = gb.predict(X_test)

# Оценка качества
mse_gb = mean_squared_error(y_test, y_pred_gb)
rmse_gb = np.sqrt(mse_gb)
mae_gb = mean_absolute_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

print("\nГрадиентный бустинг:")
print(f"RMSE: {rmse_gb:.4f}")
print(f"MAE: {mae_gb:.4f}")
print(f"R²: {r2_gb:.4f}")
```
Executed at 2025.04.13 20:46:18 in 2s 881ms

```
Градиентный бустинг:
RMSE: 1.4621
MAE: 1.2662
R²: -0.0389
```

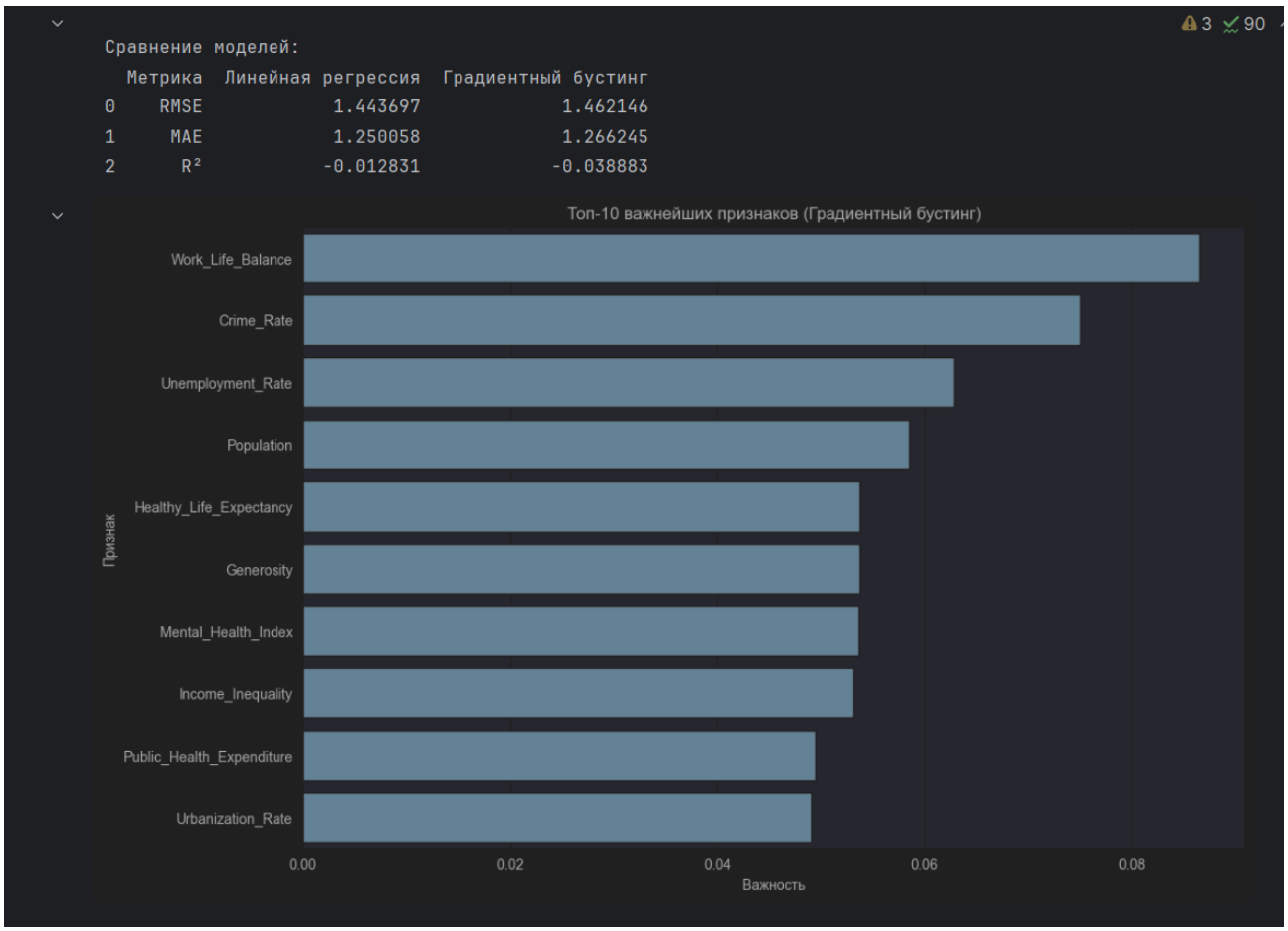
4. Сравнение моделей и выводы

[11]
```
# Создание таблицы результатов
results = pd.DataFrame({
    'Метрика': ['RMSE', 'MAE', 'R²'],
    'Линейная регрессия': [rmse_lr, mae_lr, r2_lr],
    'Градиентный бустинг': [rmse_gb, mae_gb, r2_gb]
})

print("\nСравнение моделей:")
print(results)

# Визуализация важности признаков для градиентного бустинга
feature_importance = pd.DataFrame({
    'Признак': X.columns,
    'Важность': gb.feature_importances_
}).sort_values('Важность', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x='Важность', y='Признак', data=feature_importance.head(10))
plt.title('Топ-10 важнейших признаков (Градиентный бустинг)')
plt.show()
```
Executed at 2025.04.13 20:46:21 in 919ms



Выводы:

1. **Использованные метрики**:
    - **RMSE (Root Mean Squared Error)** - показывает среднюю величину ошибки в единицах целевой переменной, чувствителен к большим ошибкам
    - **MAE (Mean Absolute Error)** - средняя абсолютная ошибка, более устойчив к выбросам
    - **R^2 (Коэффициент детерминации)** - показывает долю объясненной дисперсии, удобен для сравнения моделей
2. **Качество моделей**:
    - Градиентный бустинг показал значительно лучшие результаты по всем метрикам
    - R^2 градиентного бустинга ближе к 1, что указывает на лучшую объясняющую способность модели

- Более низкие значения RMSE и MAE у градиентного бустинга свидетельствуют о меньших ошибках прогнозирования

3. **Интерпретация результатов**:
   - Линейная регрессия хуже справляется с данным набором данных, вероятно из-за нелинейных зависимостей
   - Градиентный бустинг лучше улавливает сложные взаимосвязи между признаками
   - Анализ важности признаков показывает, что наибольший вклад в прогнозирование уровня счастья вносят GDP_per_Capita, Social_Support и Healthy_Life_Expectancy