МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт кибербезопасности и цифровых технологий (ИКБ)

КБ-2 «Информационно-аналитические системы кибербезопасности»

ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №5

В РАМКАХ ДИСЦИПЛИНЫ «МЕТОДЫ АНАЛИЗА ДАННЫХ»

Выполнил:

Студент 4-ого курса

Учебной группы БИСО-02-22

Зубарев В.С.

Москва 2025

In [4]:

```python
# Парктика 1
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import r2_score

# Загрузка данных
data = pd.read_csv('abalone.csv')

# Преобразование признака Sex
data['Sex'] = data['Sex'].map(lambda x: 1 if x == 'M' else (-1 if x == 'F' else

# Разделение на признаки и целевую переменную
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Настройка кросс-валидации
kf = KFold(n_splits=5, shuffle=True, random_state=1)

# Список для хранения результатов
results = []

# Перебор количества деревьев от 1 до 50
for n_trees in range(1, 51):
    model = RandomForestRegressor(n_estimators=n_trees, random_state=1)
    r2_scores = []

    # Кросс-валидация
    for train_index, test_index in kf.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        r2_scores.append(r2_score(y_test, y_pred))

    mean_r2 = np.mean(r2_scores)
    results.append((n_trees, mean_r2))

    # Проверка условия для досрочного прерывания
    if mean_r2 > 0.52:
        print(f"Минимальное количество деревьев: {n_trees}")
        print(f"Качество (R²): {mean_r2:.4f}")
        break
else:
    print("Не удалось достичь R² > 0.52 в диапазоне 1-50 деревьев")

# Анализ изменения качества
if len(results) > 1:
    print("\nАнализ изменения качества:")
    for i in range(1, len(results)):
        prev_score = results[i-1][1]
        curr_score = results[i][1]
        if curr_score < prev_score:
            print(f"При {results[i][0]} деревьях качество ухудшилось с {prev_sco
    print("В целом качество стабилизируется после достижения определенного колич
```

```
Минимальное количество деревьев: 21
Качество (R²): 0.5205

Анализ изменения качества:
При 11 деревьях качество ухудшилось с 0.4954 до 0.4944
При 20 деревьях качество ухудшилось с 0.5198 до 0.5195
В целом качество стабилизируется после достижения определенного количества деревь
ев
```

In [ ]:

In [2]:

```
# Практика 2
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.metrics import log_loss

# 1. Загрузка данных и разделение
try:
    data = pd.read_csv('gbm-data.csv')
except FileNotFoundError:
    print("Ошибка: Файл 'gbm-data.csv' не найден в текущей директории.")
    exit(1)

X = data.iloc[:, 1:].values
y = data.iloc[:, 0].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.8, random_state=241
)

# 2. Обучение Gradient Boosting для разных learning_rate
learning_rates = [1, 0.5, 0.3, 0.2, 0.1]
results = {}

for lr in learning_rates:
    print(f"\nОбучение с learning_rate={lr}")
    gb = GradientBoostingClassifier(
        n_estimators=250,
        learning_rate=lr,
        random_state=241,
        verbose=True
    )
    gb.fit(X_train, y_train)

    # Получаем предсказания на каждой итерации
    train_staged = gb.staged_decision_function(X_train)
    test_staged = gb.staged_decision_function(X_test)

    train_losses = []
    test_losses = []

    # Вычисляем log-loss для каждой итерации
    for train_pred, test_pred in zip(train_staged, test_staged):
        # Применяем сигмоиду
        train_prob = 1 / (1 + np.exp(-train_pred))
        test_prob = 1 / (1 + np.exp(-test_pred))

        # Убеждаемся, что массивы имеют правильную форму
        if train_prob.ndim > 1:
            train_prob = train_prob[:, 0]  # Для бинарной классификации берем пе
        if test_prob.ndim > 1:
            test_prob = test_prob[:, 0]

        # Вычисляем log-loss
        train_loss = log_loss(y_train, train_prob)
        test_loss = log_loss(y_test, test_prob)
```

```
        train_losses.append(train_loss)
        test_losses.append(test_loss)

    results[lr] = {
        'train_losses': train_losses,
        'test_losses': test_losses,
        'min_test_loss': min(test_losses),
        'best_iter': np.argmin(test_losses) + 1  # +1 так как итерации начинаютс
    }

# 3. Анализ переобучения для learning_rate=0.2
test_losses_lr02 = results[0.2]['test_losses']
train_losses_lr02 = results[0.2]['train_losses']

# Проверяем, начинает ли тестовая ошибка расти при продолжающемся снижении обуча
overfitting_detected = False
for i in range(1, len(test_losses_lr02)):
    if test_losses_lr02[i] > test_losses_lr02[i-1] and train_losses_lr02[i] < tr
        overfitting_detected = True
        break

print(f"\nХарактер графика качества на тестовой выборке: {'overfitting' if overf

# 4. Результаты для learning_rate=0.2
best_iter_lr02 = results[0.2]['best_iter']
min_test_loss_lr02 = results[0.2]['min_test_loss']
print(f"\nДля learning_rate = 0.2:")
print(f"Минимальное значение log-loss на тестовой выборке: {min_test_loss_lr02:.
print(f"Номер итерации: {best_iter_lr02}")

# 5. RandomForest с количеством деревьев = best_iter_lr02
rf = RandomForestClassifier(
    n_estimators=best_iter_lr02,
    random_state=241
)
rf.fit(X_train, y_train)

# Получаем вероятности принадлежности к классу 1
rf_probs = rf.predict_proba(X_test)[:, 1]
rf_log_loss = log_loss(y_test, rf_probs)
print(f"\nLog-loss для RandomForest: {rf_log_loss:.2f}")
```

```
Обучение с learning_rate=1
      Iter       Train Loss   Remaining Time
         1           1.0190          14.09s
         2           0.9192           9.87s
         3           0.8272           8.33s
         4           0.7834           7.57s
         5           0.7109           7.14s
         6           0.6368           6.82s
         7           0.5797           6.57s
         8           0.5610           6.38s
         9           0.5185           6.23s
        10           0.4984           6.12s
        20           0.1999           5.46s
        30           0.1313           5.09s
        40           0.0790           4.79s
        50           0.0511           4.54s
        60           0.0352           4.29s
        70           0.0245           4.04s
        80           0.0162           3.81s
        90           0.0114           3.58s
       100           0.0077           3.35s
       200           0.0002           1.11s

Обучение с learning_rate=0.5
      Iter       Train Loss   Remaining Time
         1           1.1255           5.48s
         2           1.0035           5.58s
         3           0.9386           5.52s
         4           0.8844           5.45s
         5           0.8381           5.42s
         6           0.7995           5.39s
         7           0.7559           5.37s
         8           0.7205           5.34s
         9           0.6958           5.33s
        10           0.6725           5.31s
        20           0.4672           5.09s
        30           0.3179           4.88s
        40           0.2274           4.65s
        50           0.1774           4.45s
        60           0.1394           4.22s
        70           0.1050           3.99s
        80           0.0805           3.76s
        90           0.0650           3.54s
       100           0.0511           3.31s
       200           0.0058           1.10s

Обучение с learning_rate=0.3
      Iter       Train Loss   Remaining Time
         1           1.2095           5.48s
         2           1.1006           5.46s
         3           1.0240           5.43s
         4           0.9729           5.41s
         5           0.9387           5.39s
         6           0.8948           5.37s
         7           0.8621           5.35s
         8           0.8360           5.32s
         9           0.8171           5.33s
        10           0.7883           5.33s
        20           0.6164           5.08s
        30           0.4933           4.88s
```

```
        40           0.4248            4.66s
        50           0.3345            4.45s
        60           0.2760            4.23s
        70           0.2263            4.00s
        80           0.1971            3.77s
        90           0.1693            3.55s
       100           0.1388            3.33s
       200           0.0294            1.11s

Обучение с learning_rate=0.2
      Iter       Train Loss   Remaining Time
         1           1.2613            5.48s
         2           1.1715            5.58s
         3           1.1009            5.43s
         4           1.0529            5.42s
         5           1.0130            5.44s
         6           0.9740            5.41s
         7           0.9475            5.38s
         8           0.9197            5.36s
         9           0.8979            5.30s
        10           0.8730            5.28s
        20           0.7207            5.06s
        30           0.6055            4.84s
        40           0.5244            4.64s
        50           0.4501            4.43s
        60           0.3908            4.20s
        70           0.3372            3.98s
        80           0.3009            3.76s
        90           0.2603            3.54s
       100           0.2327            3.31s
       200           0.0835            1.11s

Обучение с learning_rate=0.1
      Iter       Train Loss   Remaining Time
         1           1.3199            5.48s
         2           1.2645            5.47s
         3           1.2170            5.44s
         4           1.1775            5.42s
         5           1.1404            5.39s
         6           1.1106            5.37s
         7           1.0844            5.35s
         8           1.0617            5.33s
         9           1.0411            5.31s
        10           1.0223            5.28s
        20           0.8864            5.05s
        30           0.7844            4.83s
        40           0.7176            4.62s
        50           0.6590            4.40s
        60           0.6120            4.17s
        70           0.5599            3.95s
        80           0.5242            3.74s
        90           0.4829            3.54s
       100           0.4473            3.32s
       200           0.2379            1.10s

Характер графика качества на тестовой выборке: overfitting

Для learning_rate = 0.2:
Минимальное значение log-loss на тестовой выборке: 0.53
Номер итерации: 37
```

```
Log-loss для RandomForest: 0.54
```

In [ ]: