Отчёт по практической работе №3

Ансамбли алгоритмов для решения задачи регрессии. Веб-сервер

Практикум 317 группы, ММП ВМК МГУ

2025

Содержание

1 Введение

В данной практической работе были реализованы два классических алгоритма машинного обучения для задачи регрессии: **Random Forest** (случайный лес) и **Gradient Boosting** (градиентный бустинг). Также был разработан веб-сервер с REST API и графическим интерфейсом для обучения и инференса моделей.

2 Реализация алгоритмов

2.1 Random Forest (Случайный лес)

Случайный лес — это ансамблевый алгоритм, который строит множество независимых деревьев решений и усредняет их предсказания.

Основные особенности реализации:

- **Bootstrap-сэмплирование**: каждое дерево обучается на случайной выборке с возвращением
- **Параметры**: `n_estimators`, `max_depth`, `max_features`
- **Early stopping**: остановка обучения при отсутствии улучшения метрики
- **Сериализация**: сохранение и загрузка обученных моделей через `joblib`

Формула предсказания:

$$\hat{y}(x) = \frac{1}{T} \sum_{t=1}^{T} b_t(x)$$

где T — количество деревьев, $b_t(x)$ — предсказание t-го дерева.

2.2 Gradient Boosting (Градиентный бустинг)

Градиентный бустинг строит ансамбль последовательно, где каждое следующее дерево обучается на остатках (антиградиенте) предыдущих.

Основные особенности реализации:

- **Последовательное обучение**: каждое дерево корректирует ошибки предыдущих
- **Learning rate**: коэффициент сжатия для регуляризации
- **Начальное приближение**: среднее значение целевой переменной
- **Early stopping**: аналогично Random Forest

Формула обновления:

$$F_m(x) = F_{m-1}(x) + \eta \cdot b_m(x)$$

где η — learning rate, $b_m(x)$ — дерево, обученное на остатках $y - F_{m-1}(x)$.

3 Эксперименты

3.1 Датасет

Использовался датасет **House Sales in King County, USA** — данные о продажах недвижимости.

- Размер датасета: 21613 объектов, 90 признаков (после предобработки)
- Целевая переменная: `price` (цена дома)
- Диапазон цен: от \$75,000 до \$7,700,000

Предобработка данных:

- Извлечение признаков из даты (год, месяц, день)
- One-hot encoding для zipcode (70 уникальных значений)
- Удаление идентификаторов (`id`, `date`)

Разделение данных:

- Train: 13832 объектов (64%)
- Validation: 3458 объектов (16%)
- Test: 4323 объектов (20%)

3.2 Эксперименты с Random Forest

3.2.1 Зависимость от количества деревьев

Таблица 1: Random Forest: зависимость RMSE от количества деревьев

n_estimators	**RMSE (val)**	**RMSE (test)**	**Время (сек)**
5	135,054	170,508	0.33
10	131,174	159,078	0.66
20	130,160	157,400	1.37
30	126,733	156,623	1.98
50	127,742	155,760	3.27
75	127,153	152,283	5.17
100	127,121	153,623	6.53

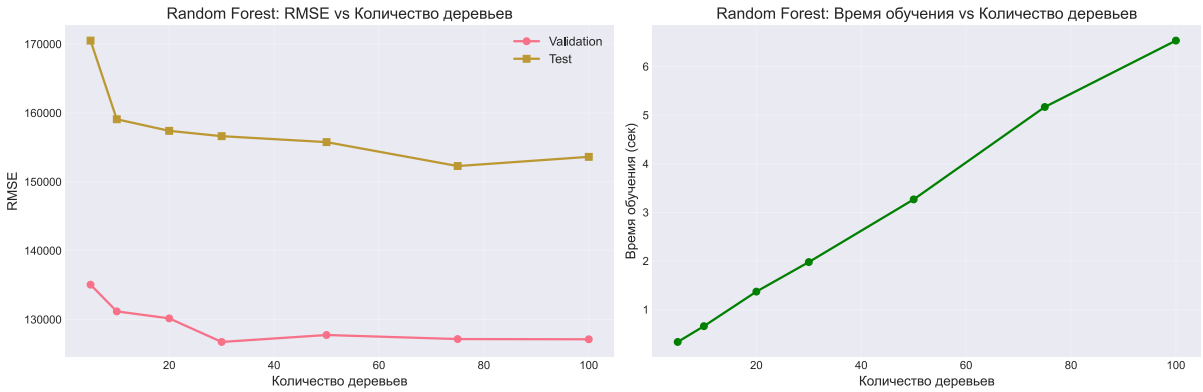

Рис. 1: Random Forest: RMSE и время обучения vs количество деревьев

Вывод: Увеличение количества деревьев улучшает качество, но рост замедляется после 50-75 деревьев. Время обучения растёт линейно.

3.2.2 Зависимость от максимальной глубины

Таблица 2: Random Forest: зависимость RMSE от максимальной глубины

max_depth	**RMSE (val)**	**RMSE (test)**	**Время (сек)**
3	205,498	230,660	0.70
5	165,025	190,881	1.10
10	127,954	160,027	1.97
15	118,806	149,638	2.68
20	122,617	150,265	3.11
None	124,106	155,718	3.26

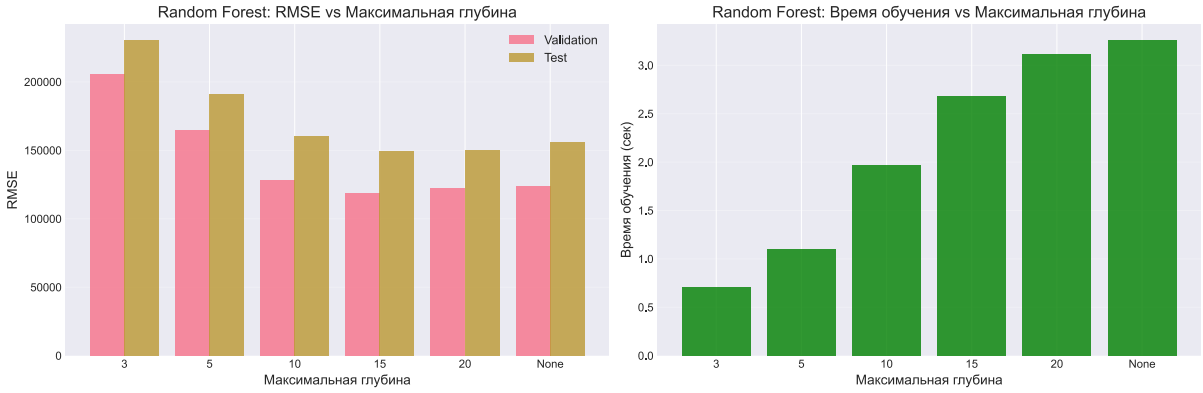

Рис. 2: Random Forest: RMSE и время обучения vs максимальная глубина

Вывод: Оптимальная глубина деревьев — 10-15. Неограниченная глубина (None) приводит к переобучению.

3.2.3 Зависимость от max_features

Таблица 3: Random Forest: зависимость RMSE от max_features

max_features	Доля	RMSE (val)	RMSE (test)	Время (сек)
9 (sqrt)	10%	153,260	178,218	0.50
22	24%	139,559	168,218	0.93
45	50%	131,505	155,031	1.76
90 (all)	100%	127,829	157,079	3.34

Вывод: Использование всех признаков даёт лучшее качество на валидации, но 50% признаков обеспечивает лучшую генерализацию на тесте при меньшем времени обучения.

3.3 Эксперименты с Gradient Boosting

3.3.1 Зависимость от количества деревьев

Таблица 4: Gradient Boosting: зависимость RMSE от количества деревьев

n_estimators	RMSE (val)	RMSE (test)	Время (сек)
10	221,208	248,028	0.23
20	173,331	194,330	0.45
30	152,223	171,298	0.71
50	133,175	152,932	1.19
75	125,540	145,145	1.79
100	121,154	140,531	2.39
150	116,847	135,827	3.63

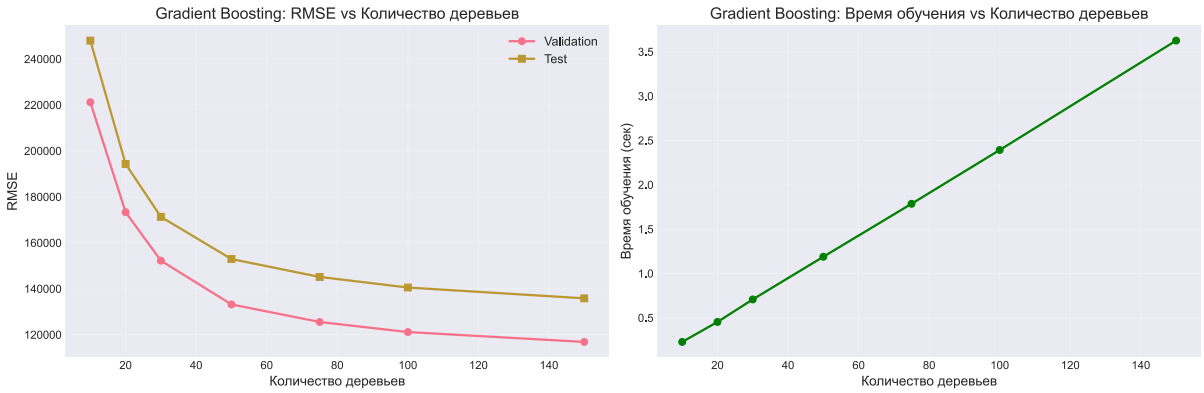

Рис. 3: Gradient Boosting: RMSE и время обучения vs количество деревьев

Вывод: Gradient Boosting более чувствителен к количеству деревьев — качество продолжает улучшаться даже при 150 деревьях.

3.3.2 Зависимость от максимальной глубины

Таблица 5: Gradient Boosting: зависимость RMSE от максимальной глубины

max_depth	RMSE (val)	RMSE (test)	Время (сек)
2	155,082	181,875	0.81
3	133,175	152,932	1.17
5	116,550	145,765	1.89
7	115,900	139,932	2.60
10	116,442	146,441	3.71
None	161,434	198,069	6.71

Вывод: Оптимальная глубина базовых деревьев — 5-7. В отличие от Random Forest, неограниченная глубина сильно ухудшает качество из-за переобучения на каждом шаге.

3.3.3 Зависимость от learning rate

Таблица 6: Gradient Boosting: зависимость RMSE от learning rate

learning_rate	RMSE (val)	RMSE (test)	Время (сек)
0.01	268,341	301,634	1.19
0.05	161,282	182,117	1.14
0.10	133,175	152,932	1.15
0.20	123,159	144,301	1.14
0.30	119,960	150,495	1.16
0.50	128,727	149,342	1.15

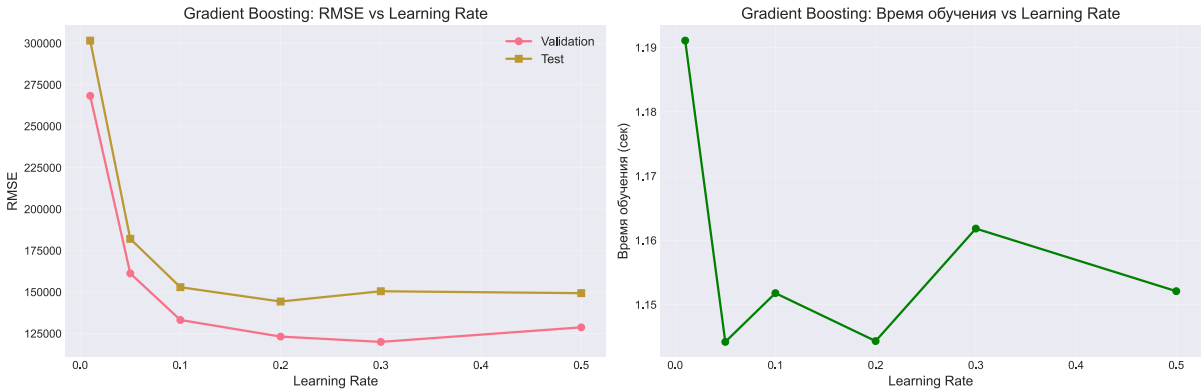

Рис. 4: Gradient Boosting: RMSE vs Learning Rate

Вывод: Оптимальный learning rate — 0.1-0.2. Слишком маленькие значения требуют больше деревьев, слишком большие (>0.3) приводят к переобучению.

3.4 Итоговое сравнение моделей

Таблица 7: Сравнение лучших конфигураций моделей

Модель	**RMSE (test)**	**Время обучения (сек)**
Random Forest (n=75, depth=15)	146,370	6.89
Gradient Boosting (n=100, depth=5, lr=0.1)	141,838	3.88

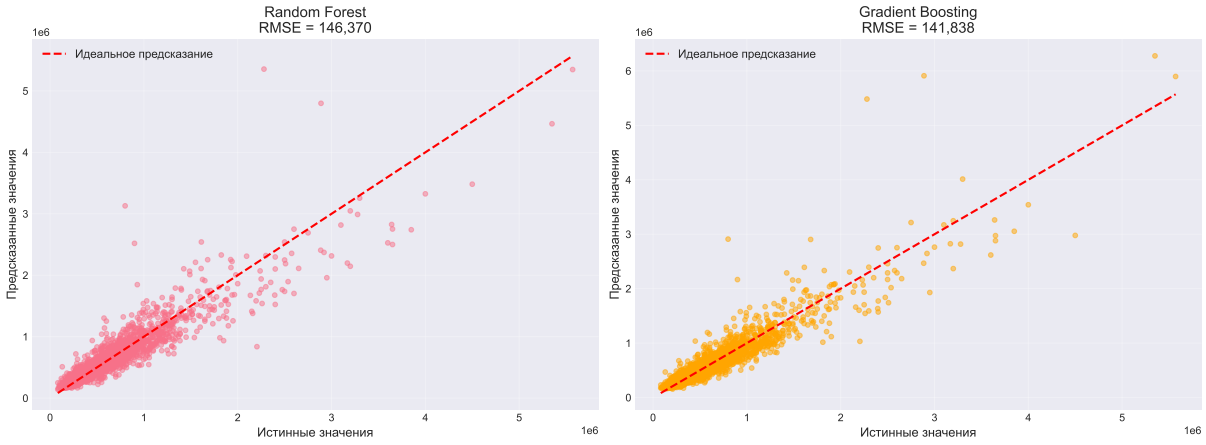

Рис. 5: Сравнение предсказаний Random Forest и Gradient Boosting

3.5 Выводы из экспериментов

Random Forest:

- Увеличение количества деревьев улучшает качество, но рост замедляется после 50-75 деревьев
- Оптимальная глубина деревьев: 10-15
- Время обучения растёт линейно с количеством деревьев
- Неограниченная глубина приводит к переобучению
- Менее чувствителен к гиперпараметрам

Gradient Boosting:

- Более чувствителен к количеству деревьев — качество продолжает улучшаться дольше
- Оптимальная глубина базовых деревьев: 3-7 (значительно меньше, чем у RF)
- Learning rate 0.1-0.2 обеспечивает хороший баланс между качеством и скоростью
- Слишком большой learning rate (>0.3) ухудшает качество
- При правильной настройке показывает лучшее качество

Общие выводы:

- Gradient Boosting показал лучший результат (RMSE = 141,838 vs 146,370)
- Gradient Boosting обучается быстрее при оптимальных параметрах
- Random Forest более устойчив к выбору гиперпараметров
- Оба алгоритма хорошо справляются с задачей предсказания цен на недвижимость

4 Веб-сервер

4.1 Архитектура

Система построена на микросервисной архитектуре с двумя компонентами:

1. **Backend** (FastAPI) — REST API на порту 8000
2. **Frontend** (Streamlit) — веб-интерфейс на порту 8501

4.2 Backend API

Реализованы следующие эндпоинты:

Метод	Endpoint	Описание
GET	/existing_experiments/	Список экспериментов
POST	/register_experiment/	Создание эксперимента
GET	/experiment_config/	Получение конфигурации
GET	/needs_training	Проверка статуса обучения
POST	/train/	Запуск обучения
GET	/convergence_history/	История сходимости
POST	/predict/	Предсказание
GET	/health	Проверка здоровья

4.3 Структура проекта

```
ensembles/              # ML
  random_forest.py      # RandomForestMSE
  boosting.py           # GradientBoostingMSE
  utils.py              # RMSLE, early stopping
  backend.py            # ExperimentConfig schema
  frontend.py           # HTTP Client

backend/                # FastAPI
  ml_app.py             #
  src/experiments/      # API
    schemas.py          # Pydantic
    service.py          #             -
    router.py           #
```

```
ui.py                    # Streamlit
runs/                    #
```

5 Docker

Приложение упаковано в Docker с использованием `docker-compose`:

```
services:
  backend:
    build: backend/Dockerfile
    ports: 8000:8000
    healthcheck: /health

  frontend:
    build: Dockerfile.streamlit
    ports: 8501:8501
    depends_on: backend (healthy)
```

Запуск:

```
docker-compose up -d
#
make up
```

6 Функциональность интерфейса

1. **Создание эксперимента**: выбор модели, гиперпараметров, загрузка CSV
2. **Обучение модели**: визуализация кривых обучения (train/val loss)
3. **Инференс**: загрузка тестовых данных и получение предсказаний

7 Заключение

В ходе работы были успешно реализованы:

- Алгоритмы Random Forest и Gradient Boosting с поддержкой early stopping
- Проведены эксперименты на датасете House Sales in King County
- REST API на FastAPI с полной документацией (Swagger UI)
- Веб-интерфейс на Streamlit для работы без знания Python
- Docker-контейнеризация для простого развёртывания

Gradient Boosting показал лучшее качество предсказания (RMSE = 141,838) при меньшем времени обучения. Система позволяет пользователям без навыков программирования обучать модели машинного обучения и делать предсказания через удобный веб-интерфейс.