Тестирование ML-моделей

Ирина Степановна Трубчик

https://t.me/+PsC-JDrwrvsxNmVi

Лекция 10

Цели занятия

1. Зачем тестировать ML-модели
2. Виды тестирования в ML
3. Инструменты для ML-тестирования
4. Чек-лист ML-тестирования

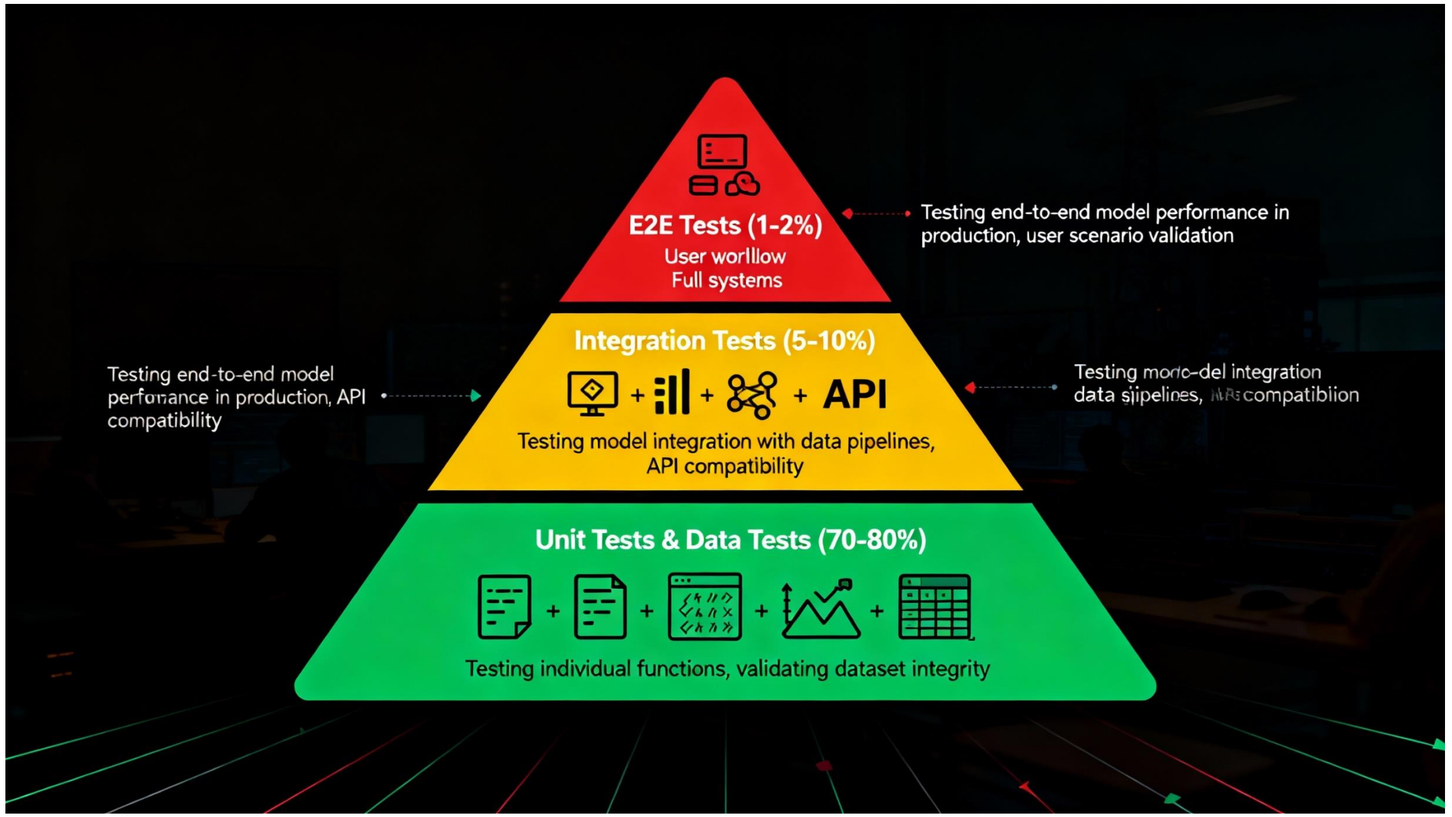
"Встречали ли вы баги в ML-коде? Как вы их обнаружили?"
"Какие типы ошибок вас больше всего беспокоят?"
"Как вы сейчас тестируете свои модели?"

зачем тестировать ML-модели

Тестирование — ключ к производственным ML-системам. Даже самая «умная» модель может совершать ошибки на новых данных, а баг в тестовой логике может стать причиной потери денег/доверия.

Приведите примеры, когда баг в ML-модели стоил дорого.

Пирамида тестирования в ML



Unit Tests & Data Tests (70-80% тестов)
Самые быстрые и дешевые тесты. Проверяют отдельные компоненты.
Unit Tests:

Функции для нормализации, оценки метрик
Преобразование признаков; утилиты обработки данных

Data Tests
проверка целостности данных; валидация схемы; поиск аномалий

Виды тестирования в ML

- Data tests (валидация, очистка)
- Unit tests (модули препроцессинга)
- Integration tests (поток данных)
- Model tests (метрики качества)
- System tests (end-to-end сквозной тест)

Какие из этих тестов вы уже используете?

Тестирование данных до обучения

- Проверка пропусков, выбросов, некорректных значений
- Great Expectations, Pandera (Python)
- Примеры: expect_column_values_between, expect_column_values_to_not_be_null

Какие ошибки с данными встречали на практике?

Unit-тесты для функций предобработки

- Проверка конкретных функций (normalize, one-hot, заполнение NA)
- Использование pytest или unittest

Пример: test_preprocessing_handles_missing_values

```
def test_normalizer():
    assert normalize([1,2,3]) == [0.0,0.5,1.0]

Или тест на пропуски (запуск кода, код – в конспекте)
pytest test_preprocessing.py -v
```

Проверка стабильности train/test split

- Проверка stratisfied split/утечки
- Distribution of labels/features: does train ~ test?
- Визуализация: pairplot, histogram

Заметили ли вы когда-то утечку данных при split?

иллюстрация проверки стабильности для модели

Описание диаграммы:

Эта иллюстрация показывает процесс тестирования robustness (устойчивости) ML-модели:

1. Clean Data (Чистые данные) — исходное распределение тестовых данных
2. Add Noise (Добавление шума) — данные с гауссовским шумом (небольшие случайные возмущения)
3. Compare Predictions (Сравнение предсказаний) — проверка что предсказания модели не сильно изменились

Ключевая метрика: Mean difference < 0.1 — средняя разница между предсказаниями на чистых и зашумленных данных должна быть небольшой.

Пример кода для этого теста:

```
def test_model_robust_to_noise():
    """Проверка устойчивости модели к шуму"""

    # Чистые данные
    X_test_clean = X_test.copy()

    # Добавляем гауссовский шум (σ = 0.1)
    noise = np.random.normal(0, 0.1, X_test_clean.shape)
    X_test_noisy = X_test_clean + noise

    # Получаем предсказания
    pred_clean = model.predict_proba(X_test_clean)[:, 1]
    pred_noisy = model.predict_proba(X_test_noisy)[:, 1]

    # Вычисляем разницу
    mean_diff = np.abs(pred_clean - pred_noisy).mean()

    # Модель должна быть устойчива
    assert mean_diff < 0.1, f"Model too sensitive: {mean_diff:.3f}"
```

Модульные тесты для модели

- Проверка predict() на тип данных и размерность
- Проверка, что predict(X) возвращает значения в допустимом диапазоне
- Тест предсказания "simple/edge case" (например, все null — что выдаёт?)

Тестирование метрик качества

- **Accuracy, F1, ROC AUC — тестировать на holdout, кросс-валидации**
- **Baseline — accuracy always predict most frequent class**
- **Проверка модели на переобучение (train ≫ test → overfit)**

придумайте baseline для задачи регрессии

Проверка устойчивости (robustness) модели

- Add noise/noise-attack, dropout в признаки — насколько падает accuracy?
- Adversarial examples
- Проверка на out-of-distribution данные

Почему важно тестировать на подобных примерах?

Интеграционные тесты: сквозной pipeline

- Пример: тестируется не только модель, а и всё: ingest → preprocess → inference → output
- Тестовые дата-сеты встроены (fixtures)
- Примеры: test end-to-end on small batch of real data

Regression тесты (предотвращение деградации)

- Проверка: новая модель/код не ухудшает метрики
- Автоматизация: сохранять метрики предыдущей версии
- Примеры: модели A и B — их тесты сравнивают метрики

ML специфические анти-паттерны

- ✓ *Data leakage (утечки будущей информации)*
- ✓ *Target leakage (целевое значение попадает в признаки)*
- ✓ *Overfitting на test set (подгонка к тесту)*
- ✓ *Недостаточное покрытие тестов edge-cases*

Использование CI/CD для тестирования

- GitHub Actions, GitLab CI, Jenkins: проверить все тесты при каждом push
- Проверять качество модели автоматически (test fail — нельзя в master)
- Пример config для pytest+coverage

Инструменты для ML-тестирования

- *Pytest, Unittest (Python стандарт)*
- *Great Expectations, Pandera — проверки данных*
- *MLflow — трекинг метрик, регресс- и интеграционные тесты*
- *Deepchecks (open-source), Evidently — auto-тесты и метрики*

Чек-лист ML-тестирования

- Data quality tests перед train и перед prod
- Unit tests для препроцессинга и вспомогательных функций
- Metics tests (holdout, CV)
- Regression тесты при каждом релизе модели
- E2E тест и интеграция в CI

Мониторинг — живой процесс: алерты и метрики нужно регулярно дорабатывать и совершенствовать!

Ключевые вопросы для самопроверки:

1. Почему unit-тесты важны в ML, если модель обучена на валидационном наборе?
2. Назовите три типа data leakage и как их выявить.
3. Какая разница между regression-тестом и integration-тестом?
4. Приведите пример edge-case'а которого часто забывают тестировать
5. Объясните пирамиду тестирования ML и почему 70-80% должны быть unit-тесты?
6. Какие инструменты использовать для тестирования ML в production и почему?

Домашнее задание

- *Добавить к своему проекту 2 unit-теста и 1 regression-тест.*
- *Попробовать Great Expectations или Deepchecks для проверки одного датасета.*

Материалы и ссылки

Great Expectations: https://greatexpectations.io/

Deepchecks: https://deepchecks.com/open-source/

Pandera: https://pandera.readthedocs.io/en/stable/

MLflow: https://mlflow.org/

CI/CD для ML: https://madewithml.com/courses/mlops/testing/

Книга: “Machine Learning Design Patterns” (Google, 2020), главы о тестировании

Статья: “Руководство по тестированию ML” (mlops.community) https://mlops.community/ml-test-guide/

Вопросы

Телеграм https://t.me/+PsC-JDrwrvsxNmVi

СКИФ (https://do.skif.donstu.ru/course/view.php?id=7508)