Валидация данных в ML-проектах

Дмитрий Колодезев
ООО Промсофт, Новосибирск
23.07.2020

План

- Жизненный цикл модели
- Валидация при инференсе
- Pydantic + FastAPI
- Ошибочные входные данные
- Валидация при обучении
- Pandera
- Валидация в мониторинге
- Снова про жизненный цикл

Жизненный цикл модели

- Подготовка данных
- Обучение модели
- Верификация модели
- Инференс
- Мониторинг модели
- Повторить

Валидация:инференс

- Rest API
  - Проверяем входные данные
- Библиотечная функция
  - Проверяем входные данные!
- Тип
- Диапазон
- Выбор из списка
- Соблюдение условий

Pydantic

- pip install pydantic
- Аннотация типов
- PyCharm
- MyPy
- Быстрый
- Легкий
- Простой
- Расширяемый

```python
from datetime import datetime
from typing import List, Optional
from pydantic import BaseModel


class User(BaseModel):
    id: int
    name = 'John Doe'
    signup_ts: Optional[datetime] = None
    friends: List[int] = []


external_data = {
    'id': '123',
    'signup_ts': '2019-06-01 12:22',
    'friends': [1, 2, '3'],
}
user = User(**external_data)
```

FastAPI

```python
from typing import Optional

from fastapi import FastAPI

app = FastAPI()


@app.get("/items/")
async def read_items(q: Optional[str] = None):
    results = {"items": [{"item_id": "Foo"}, {"item_id": "Bar"}]}
    if q:
        results.update({"q": q})
    return results
```

Причина страдания

- Данные откуда-то берутся:
  - Запросы из UI
  - Выгрузка из баз
  - Внешние API
  - Импорт из внешних источников
- Там все меняется
  - Валидация
  - Дефолтное значение в форме
  - Пороги, чекпоинты

А если они…

- Пришли неверные данные:
  - Ответить ошибкой (отбросить)
  - Заменить на дефолтные (обработать)
- Это бизнес-решение, у бизнеса и спросите
- Мы не можем знать, как лучше
- Бывают обстоятельства:
  - Миграция на другого поставщика данных
  - Миграция UI или схемы данных

А, давайте отбросим

- Возвращаем
- Выбрасываем исключение из библиотеки
- Пишем в Sentry

```
async def request_validation_exception_handler(
    request: Request, exc: RequestValidationError
) -> JSONResponse:
    return JSONResponse(
        status_code=HTTP_422_UNPROCESSABLE_ENTITY,
        content={"detail": jsonable_encoder(exc.errors())},
    )
```

А, давайте обработаем

- Вернуть NA
- Дефолтное значение
- Написать в Sentry
- Свой валидатор

Нестыковочка

- Разные валидаторы для обучения и инференса
- Не совпадает распределение (страдает модель)
- Идеально — валидировать тем же кодом

Гипотез настроим

- Проверка распределений признаков
  - Среднее в диапазоне ...
  - Мода
  - Максимумы и минимумы
- Проверка распределения целевой переменной
  - KLD
- Гипотезы берутся из EDA и интерпретации
- Проверять наиболее важное (SHAP + эксперты)

Pandera

- pip install pandera
- Валидация схемы
- Проверка гипотез
- Как же FastAPI?
- И этот, pydantic
- Оба два
- Нестыковка ;-(

```python
import pandas as pd
import pandera as pa

from pandera import Check, DataFrameSchema, Column, Hypothesis

df = pd.DataFrame({
    "height": [5.6, 7.5, 4.0, 7.9],
    "group": ["A", "B", "A", "B"],
})

schema = DataFrameSchema({
    "height": Column(
        pa.Float, Hypothesis.two_sample_ttest(
            "A", "B",
            groupby="group",
            relationship="less_than",
            alpha=0.05
        )
    ),
    "group": Column(pa.String, Check(lambda s: s.isin(["A", "B"])))
})

schema.validate(df)
```

Google что-то хочет сказать

- https://github.com/tensorflow/data-validation
- Статистики на данных
- Инференс и проверка схемы
- Визуализация статистик
- Детектирование аномалий
- Сдвиг и дрейф (skew and drift)
- При обучении и инференсе

TFDV: пример

```
schema = tfdv.infer_schema(statistics=train_stats)

# Compute stats for evaluation data
eval_stats = tfdv.generate_statistics_from_csv(data_location=EVAL_DATA)

# Compare evaluation data with training data
tfdv.visualize_statistics(lhs_statistics=eval_stats, rhs_statistics=train_stats,
                          lhs_name='EVAL_DATASET', rhs_name='TRAIN_DATASET')

# Check eval data for errors by validating the eval data stats using the previously inferred schema.
anomalies = tfdv.validate_statistics(statistics=eval_stats, schema=schema)
tfdv.display_anomalies(anomalies)

# Add skew comparator for 'payment_type' feature.
payment_type = tfdv.get_feature(schema, 'payment_type')
payment_type.skew_comparator.infinity_norm.threshold = 0.01

# Add drift comparator for 'company' feature.
company=tfdv.get_feature(schema, 'company')
company.drift_comparator.infinity_norm.threshold = 0.001

skew_anomalies = tfdv.validate_statistics(train_stats, schema,
                                          previous_statistics=eval_stats,
                                          serving_statistics=serving_stats)

tfdv.display_anomalies(skew_anomalies)
```

Мониторинг

- Сохраняем запросы к модели
- Сохраняем ответы модели
- Проверяем распределения в запросах
- Проверяем распределение ответов
- Алерт, ~~дизлайк, отписка,~~ переобучение

Есть же еще что-то

- https://github.com/pyeve/cerberus
- https://github.com/Julian/jsonschema
- https://github.com/keleshev/schema
- https://github.com/schematics/schematics
- https://github.com/podio/valideer
- https://github.com/alecthomas/voluptuous
- Наверное, их много

Опять про это

- Подготовка данных
- Обучение модели
- Верификация модели
- Инференс
- Мониторинг модели
- Повторить

Вопросы

Слайды тут

dkolodezev

promsoft

dkolodezev

d_key

dmitry_kolodezev

https://kolodezev.ru/download/slides-validation.pdf