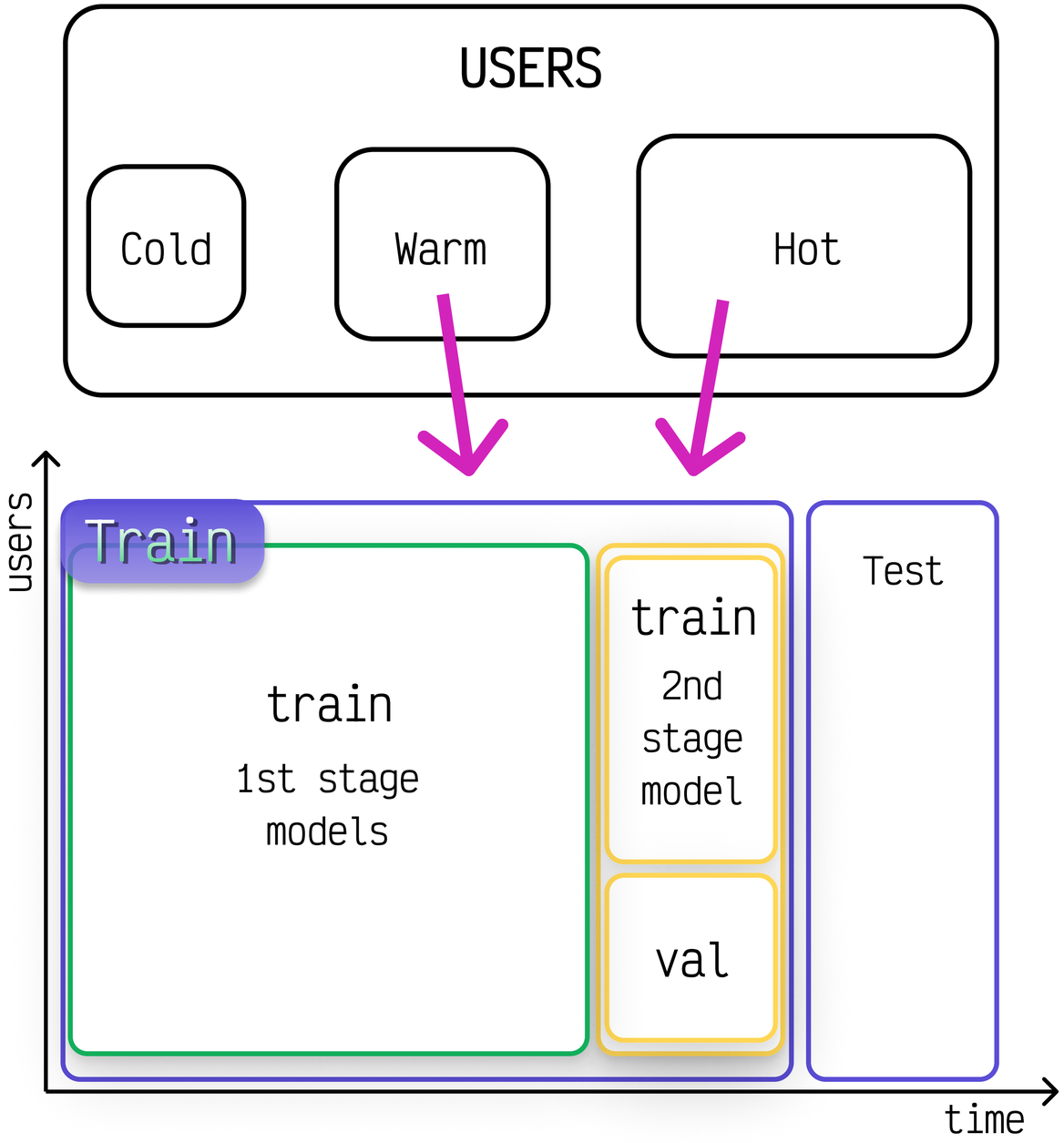
RecSys

Stage_6



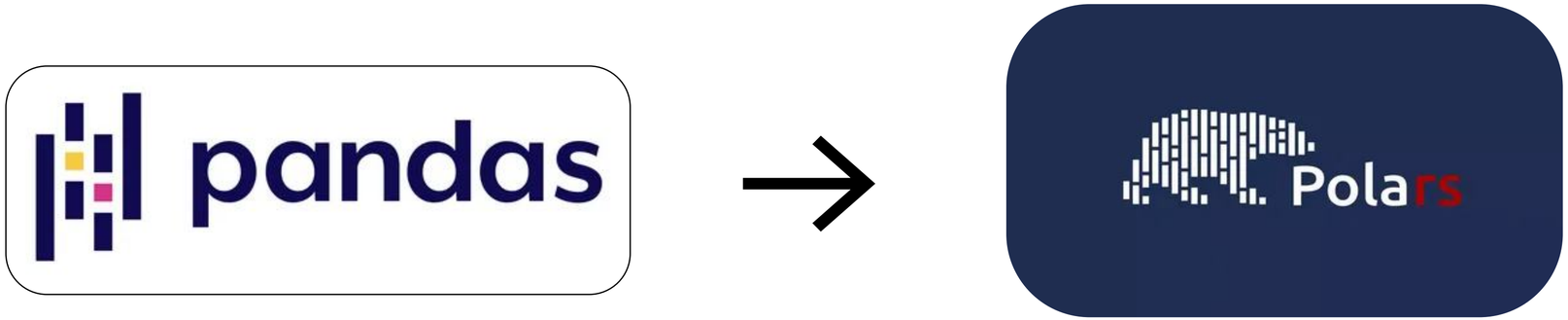
Валидация

При использовании Time-based валидации, естественно, что появятся Cold-users. Так как у нас отсутствует информация о пользователях, то для холодного старта будет использоваться модель TopPopular.

На оставшихся пользователях и будет обучаться наша модель.

Data Processing

Переход от Pandas к Polars



Выигрыш по памяти и скорости выполнения операций:

Потребление RAM:

- pandas: avg ~ 8 GB, max = 14.5 GB
- polars: avg ~ 4 GB, max = 9 GB

Время выполнения по самой затратной операции:

- pandas: ~ 5 min
- polars: ~ 1.5 min

Модели

Модели

Модели первого уровня

- KNN (cosine, BM25, TD-IDF)
- LightFM

Модели второго уровня

- CatBoost Ranker

Модели cold users

- TopPopular(topk=50) + BanditRecommender

Метрики

- Recall@k
- NDCG@k
- MAP@k

Время

Таргитированные рекомендации

Этап	Время (мин)
Треннировка + инференс моделей 1 уровня (для трен. ранкера)	15
Обработка результатов инференса моделей 1 уровня	1
Треннировка ранкера	4
Треннировка + инференс моделей 1 уровня (для инф. ранкера)	18
Инференс ранкера	1.5
ИТОГО	**40**

Холодные рекомендации

Этап	Время
Треннировка + инференс популярной модели	10 секунд
Треннировка Бандита	≤ 1 минута
Инференс Бандита	1 минута
ИТОГО	**≤ 2 минуты 10 секунд**

Модель второго уровня (метрики расчитаны на тествых данных)

ДО:

k = 3: {'ndcg@k': 0.2066009927816457, 'recall@k': 0.12447302675108117, 'map@k': 0.14127237787969413}

k = 5: {'ndcg@k': 0.24092300833035926, 'recall@k': 0.14942065114748218, 'map@k': 0.13471953670207448}

k = 10: {'ndcg@k': 0.2951177111158031, 'recall@k': 0.19172486787876636, 'map@k': 0.13414947362623275}

k = 15: {'ndcg@k': 0.3262363181849842, 'recall@k': 0.2175279962499981, 'map@k': 0.1352803591465634 2}

ПОСЛЕ:

k = 3: {'ndcg@k': 0.23283668145585287, 'recall@k': 0.1507586238930032, 'map@k': 0.16838806967687284}

k = 5: {'ndcg@k': 0.2672899635214155, 'recall@k': 0.1784753701591795, 'map@k': 0.16011627018530403}

k = 10: {'ndcg@k': 0.3140544182833436, 'recall@k': 0.2180606076346847 7, 'map@k': 0.1566453522770767}

k = 15: {'ndcg@k': 0.3387634076939192, 'recall@k': 0.2403967620639247 4, 'map@k': 0.1565731543568892}

*оставил CatBoost тюнится optuna’ой на ~5 часов

как следствие, уменьшил число итераций ⇒ быстрее инференс:
время сократилось с 5 минут до 2 минут

Взаимодействие

Докер образ

3.12 GB / 5.13 GB in use 1 images

Last refresh: 44 minutes ago

Search

Name	Tag	Created	Size	Actions	Image ID
stage_6-app	latest	39 minutes ag	4.58 GB		d7452d91bf54

dockerfiles > api.Dockerfile > ...

```dockerfile
# Use an official Python image as a parent image
FROM python:3.11.9

# Install necessary dependencies (including poetry)
RUN apt-get update && apt-get install -y --no-install-recommends apt-utils
RUN apt-get -y install curl
RUN apt-get install libgomp1

# Set the working directory in the container
WORKDIR /app

# Copy project files into the container
COPY pyproject.toml poetry.lock ./

# Upgrade pip and install poetry
RUN pip install --upgrade pip setuptools wheel
RUN pip install poetry

# Install project dependencies
RUN poetry config virtualenvs.create false && \
    poetry install --no-interaction --no-ansi --no-root -vvv && \
    poetry run pip install --upgrade pip setuptools wheel && \
    poetry run pip install lightfm && \
    poetry cache clear pypi --all -n


# Copy the rest of the project files (including the source code)
COPY . /app

# Set the entrypoint
CMD ["poetry", "run", "uvicorn", "src.api.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

docker-compose.yml

Run All Services

```yaml
services:
  Run Service
  app:
    build:
      context: .
      dockerfile: ./dockerfiles/api.Dockerfile
    ports:
      - "8000:8000"
    volumes:
      - .:/app
      - ./data:/data
```

Использован пакетный менеджер poetry

```
]
readme = "README.md"
requires-python = "=3.11.9"
dependencies = [
    "fastapi (>=0.115.6,<0.116.0)",
    "uvicorn (>=0.34.0,<0.35.0)",
    "orjson (>=3.10.14,<4.0.0)",
    "pydantic-settings (>=2.7.1,<3.0.0)",
    # "torch (>=2.5.1,<3.0.0)",
    # "torchvision (>=0.20.1,<0.21.0)",
    "pyarrow (>=18.1.0,<19.0.0)",
    "dill (>=0.3.9,<0.4.0)",
    # "transformers (>=4.48.0,<5.0.0)",
    "streamlit (>=1.41.1,<2.0.0)",
    "rectools (>=0.9.0,<0.10.0)",
    "polars (>=1.20.0,<2.0.0)",
    "scikit-learn (>=1.6.1,<2.0.0)",
    "catboost (>=1.2.7,<2.0.0)",
    "pylint (>=3.3.4,<4.0.0)",
    "mab2rec (>=1.3.1,<2.0.0)"
```

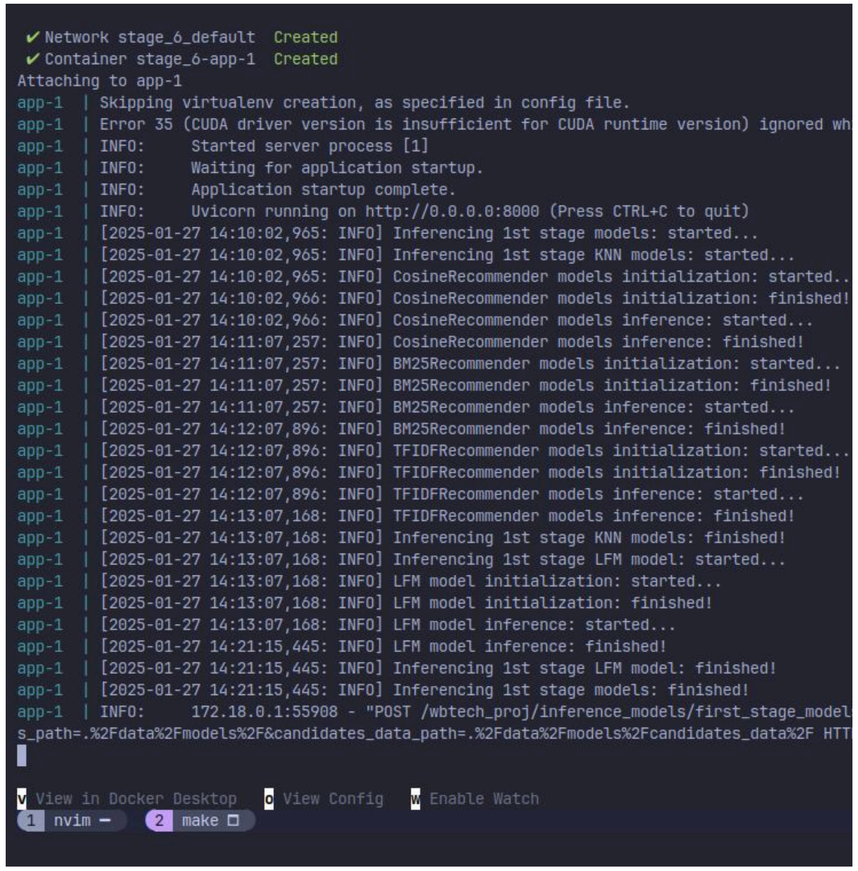
codestyle проверен pylint

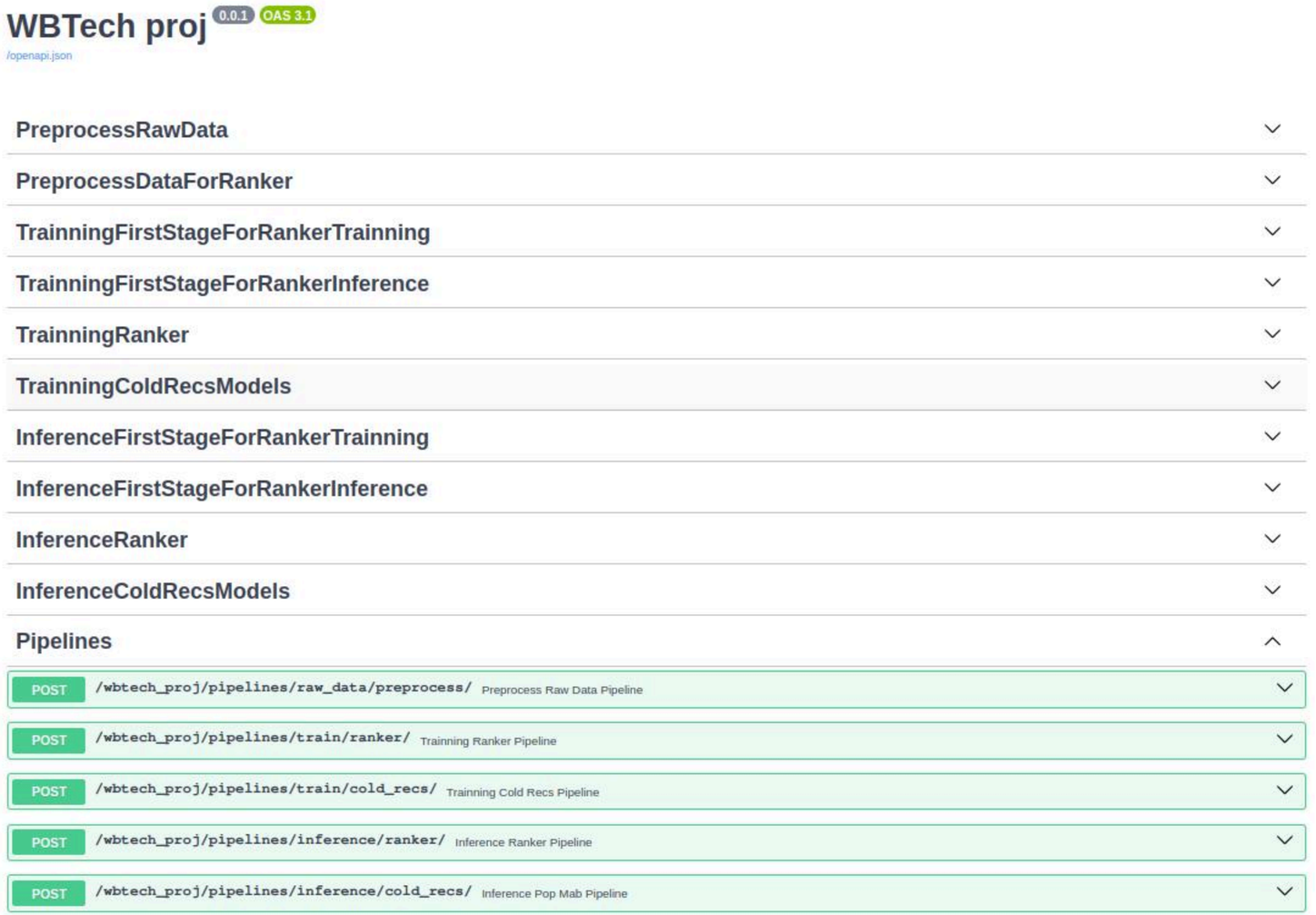
```
❯ poetry run pylint ./src/**/*.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)


~/Code/wb/recsys_project/stage_6 main*
❯
```

Реализован API (фреймворк FastAPI)

Есть ручки как для отдельных этапов решения, так и пайплайны

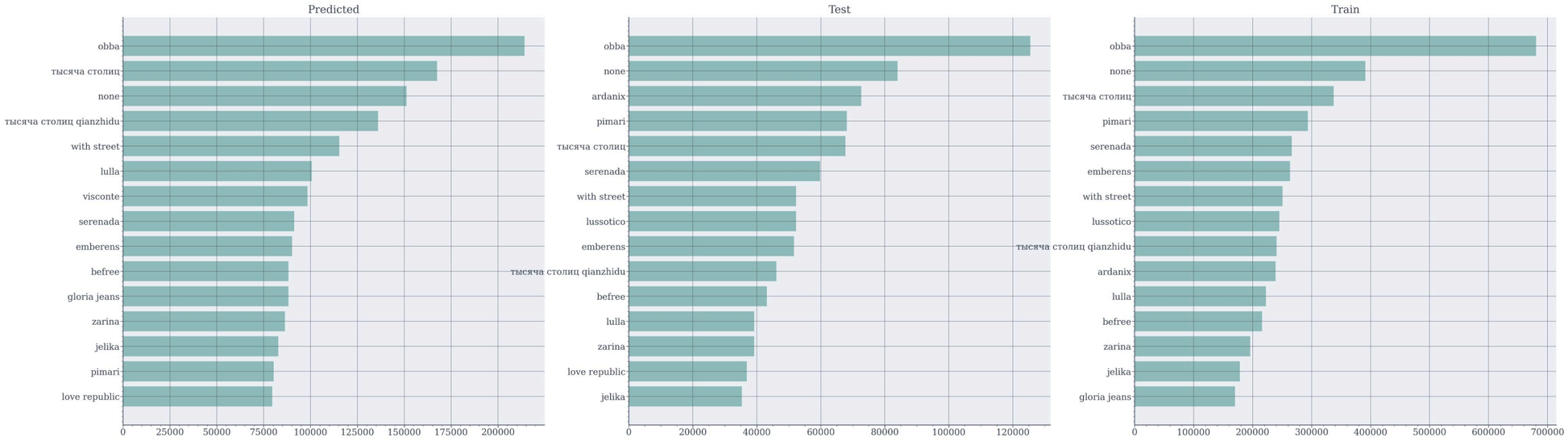
Логирование



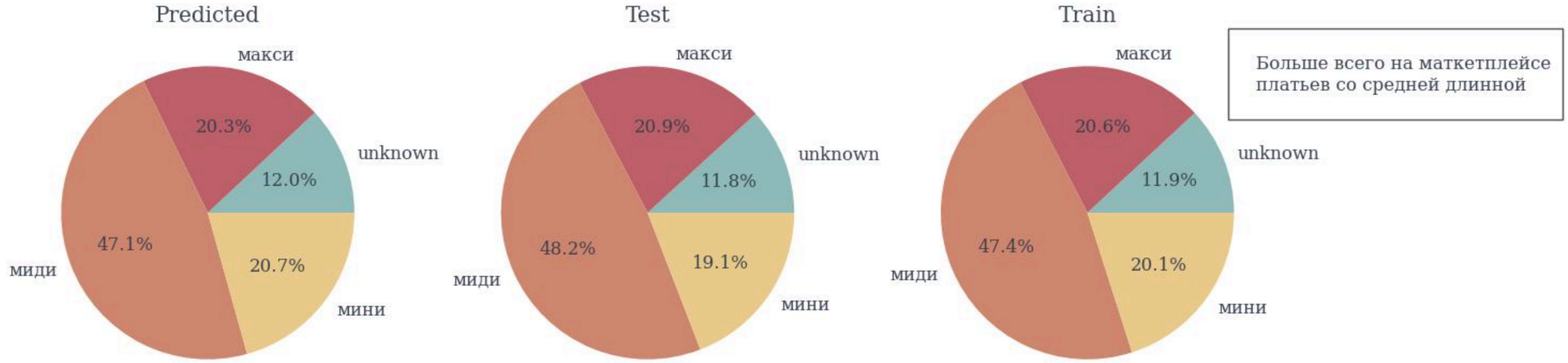


Анализ результатов

Топ популярных брендов



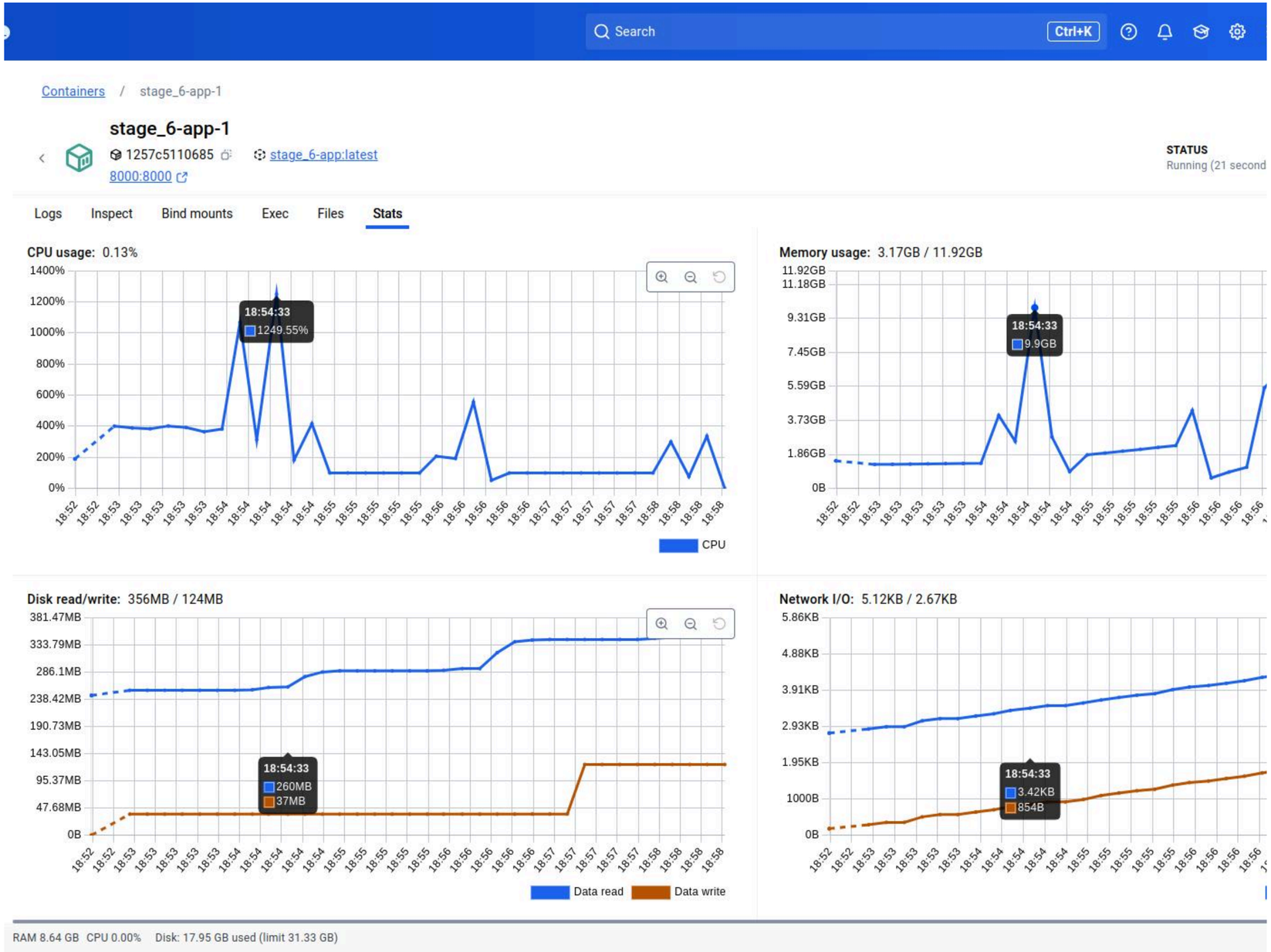
Длина платья



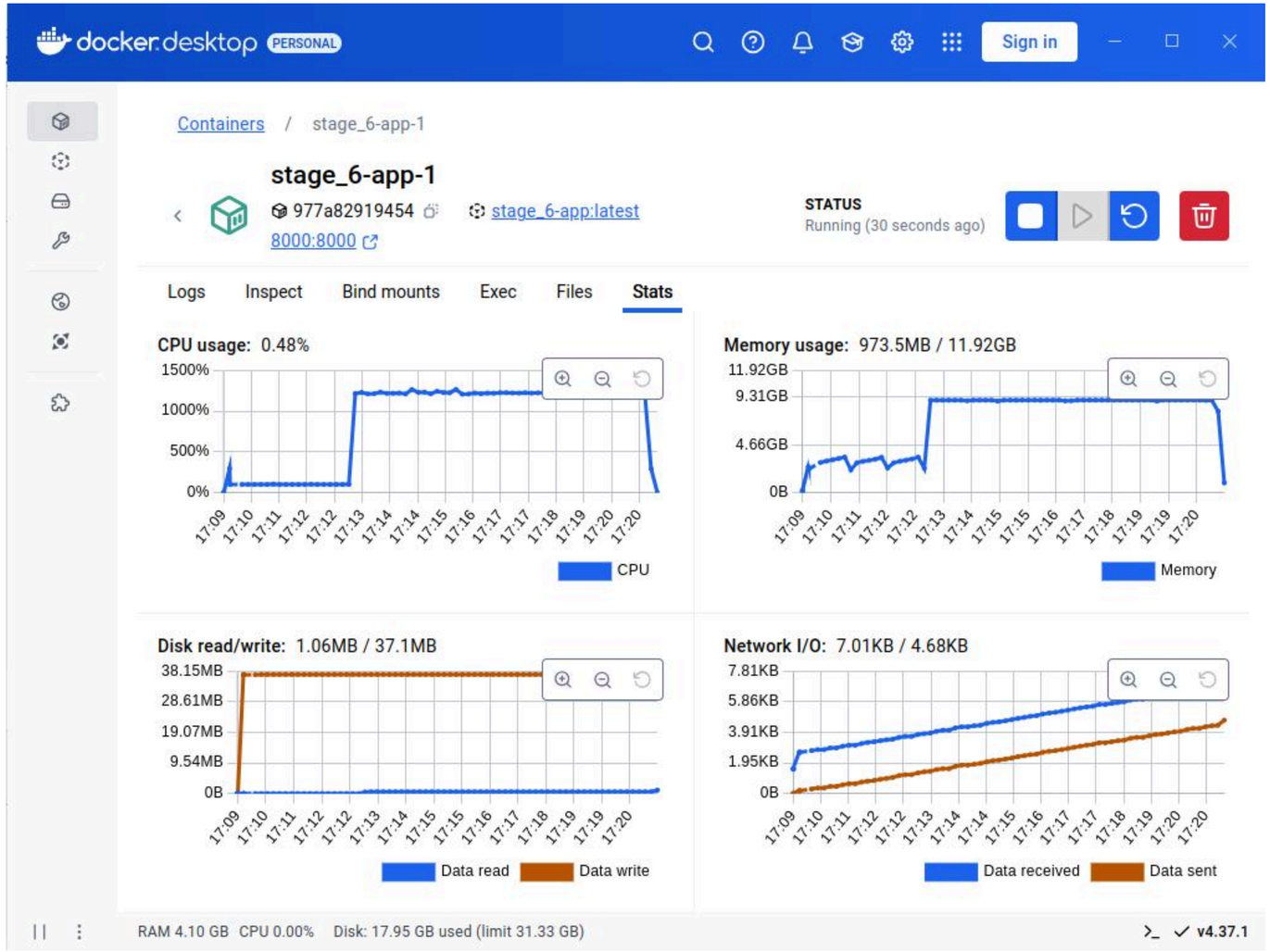
Больше всего на маткетплейсе платьев со средней длинной

Приложение

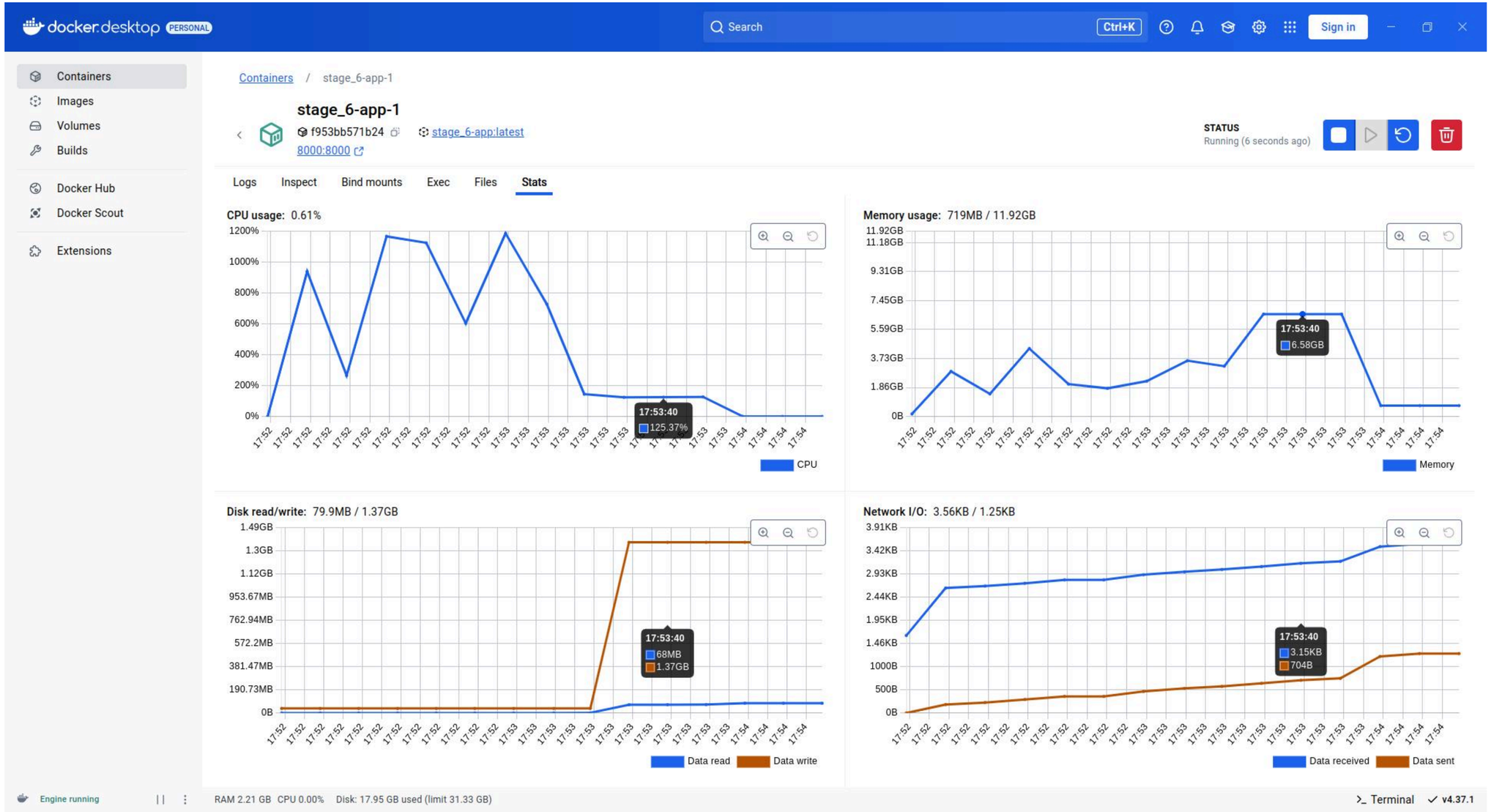
Предобработка данных



Обучение моделей моделей 1 уровня



Обработка данных моделей 1 уровня



Обучение ранкера