Министерство науки и высшего образования Российской Федерации

Национальный исследовательский университет ИТМО

Инфраструктура больших данных

Весна

2025

Лабораторная работа №2

ВЗАИМОДЕЙСТВИЕ С ИСТОЧНИКАМИ ДАННЫХ

Цель работы:

Получить навыки выгрузки исходных данных и отправки результатов модели с использованием различных источников данных согласно варианту задания.

Ход работы:

1. Создать репозитории-форк модели на GitHub, созданной в рамках лабораторной работы №1, регулярно проводить commit + push в ветку разработки, важна история коммитов.
2. Реализовать взаимодействие сервиса модели и базы данных, согласно варианту задания.
3. Обеспечить процессы аутентификации/авторизации при обращении сервиса модели к базе данных в момент отправки результата работы модели. В исходном коде не должно быть явно прописаны пары логин/пароль, адрес/порт сервера базы данных, токены доступа.
4. Возможно наполнить базу данных наборами для обучения/валидации модели.
5. Переиспользовать CI pipeline (Jenkins, Team City, Circle CI и др.) для сборки docker image и отправки их на DockerHub.
6. Переиспользовать CD pipeline для запуска контейнеров и проведения функционального тестирования по сценарию, запуск должен

стартовать по требованию или расписанию или как вызов с последнего этапа CI pipeline.

7. Результаты функционального тестирования и скрипты конфигурации CI/CD pipeline приложить к отчёту

Результаты работы:

1. Отчёт о проделанной работе;
2. Ссылка на репозиторий GitHub;
3. Ссылка на docker image в DockerHub;
4. Актуальный дистрибутив модели в zip архиве.

Обязательно использование docker-compose.

Отчет

1. Реализация обертки над базой данных Redis (src/database.py). Хранение секретов в .env файле без версирования на удаленном репозитории. Загрузка секретов осуществляется в конструкторе обертки через библиотеку python-dotenv. Для связи с базой данных используем реализацию готового Redis клиента из библиотеки redis.

Листинг 1 - Конструктор обертки над базой данных Redis.

```
import os
from redis import Redis, exceptions
from dotenv import load_dotenv
from logger import Logger

SHOW_LOG = True

class RedisClient:
     def __init__(self):
     logger = Logger(SHOW_LOG)
     self.log = logger.get_logger(__name__)
```

```
    load_dotenv(override=True)

    self.client = Redis(
          host=os.getenv('REDIS_HOST'),
          port=int(os.getenv('REDIS_PORT')),
          db=int(os.getenv('REDIS_DB')),
          password=os.getenv('REDIS_PASSWORD'),
          decode_responses=True
    )
    self.try_connect()
```

2. Сохранение результата предсказания в базу данных, реализация дополнительной ручки для получения результата.

Листинг 2 - Код ручки для вызова предсказания.

```
@app.post("/predict")
async def predict(input_data: PredictionInput):
   try:
     X = self.scaler.transform(pd.json_normalize(input_data.X))
     y = pd.json_normalize(input_data.y)
     score = self.model.score(X, y)
     pred = self.model.predict(X).tolist()

     prediction_id = str(uuid.uuid4())
     prediction_data = {
          "prediction": pred,
          "score": score,
          "input_data": input_data.model_dump_json()
     }
     self.database_client.set(
          prediction_id, json.dumps(prediction_data))

     return {"prediction_id": prediction_id}
   except Exception as e:
     self.log.error(traceback.format_exc())
```

```
    raise HTTPException(status_code=500, detail=str(e))
```

Листинг 3 - Код ручки для получения результата предсказания.

```
@app.get("/predictions/{prediction_id}")
async def get_prediction(prediction_id: str):
    prediction_data = self.database_client.get(prediction_id)
    if prediction_data is None:
        raise HTTPException(
            status_code=404,
            detail="Prediction not found")
    return json.loads(prediction_data)
```

3. Добавление базы данных в конфигурацию docker-compose.yml. Привязка web к redis при помощи инструкции depends_on.

Листинг 4 - Код docker-compose.yml.

```
services:
    web:
    build: .
    depends_on:
        redis:
            condition: service_healthy
    command: bash -c "
        python src/preprocess.py
        && python src/train.py
        && python src/predict.py -m LOG_REG -t func
        && coverage run src/unit_tests/test_preprocess.py
        && coverage run -a src/unit_tests/test_training.py
        && coverage run -a src/unit_tests/test_database.py
        && coverage run -a src/unit_tests/test_app.py
        && coverage report -m
        && (python src/app.py --model LOG_REG &)
        && sleep 30
        && curl -X GET http://localhost:8000/
        && curl -X POST http://localhost:8000/predict \
                -H 'Content-Type":" application/json' \
```

```
                    --data-binary @tests/test_0.json"
    ports:
        - 8000:8000
    image: zarus03/ml-big-data-lab-2:latest
    redis:
        image: redis:latest
        command: bash -c "redis-server --requirepass
$$REDIS_PASSWORD"
        ports:
            - 6379:6379
        env_file:
            - .env
        healthcheck:
            test: ["CMD", "redis-cli", "-a",
    "$$REDIS_PASSWORD", "ping"]
            interval: 1s
            timeout: 2s
            retries: 10
```

4. Дополнение CI пайплайна с копированием файла секретов внутрь контейнера.

Листинг 5 -Копирование файла секретов в сборку.

```
stage('Use .env File') {
    steps {
        withCredentials([file(
            credentialsId: 'redis_env_props',
            variable: 'ENV_FILE')]) {
            bat 'xcopy /F %ENV_FILE% %REPO_DIR%'
        }
    }
}
```