МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 3
MLOps с использованием ClearML | недельный прогноз погоды

по курсу
Инженерия данных

Группа 6232
Студент К.Р. Донец
(*подпись*)

Преподаватель Р.А. Парингер
(*подпись*)

Самара 2025

АРХИТЕКТУРА

На рисунке 1 представлен пайплайн из заглавного задания.

Рисунок 1 – Схема пайплайн

Кратко по архитектуре:

Общий гайд по докеру - https://docs.docker.com

разбита на 2 части ClearMl + FastApi server(с n8n)

MLOps платформа для оркестрации ML экспериментов(https://clear.ml/docs/latest/docs/)(https://hub.docker.com/r/allegroai/clearml)

Зачем сделано:

1. Автоматизация обучения моделей на исторических данных
2. HPO - автоматический поиск лучших параметров модели
3. Версионирование моделей, данных и экспериментов
4. Визуализация метрик обучения в реальном времени
5. Повторяемость экспериментов

Роль в системе: Обучение и отладка моделей.

Python веб-сервер с ML моделью(https://fastapi-tutorial.readthedocs.io/en/latest/)

Зачем сделано:

1. ML инференс в реальном времени (прогноз температуры)
2. REST API для интеграции с другими системами
3. Загрузка HPO-оптимизированных моделей из ClearML
4. Поддержка разных городов с разными температурными профилями

Роль в системе: делает прогнозы по запросу

N8N(https://docs.n8n.io)

Low-code платформа для автоматизации потоков (редактируемые бизнес-процессы)

Зачем сделано:

1. Автоматизация еженедельных прогнозов (каждый понедельник в 7:00)
2. Интеграция с FastAPI для получения прогнозов
3. Форматирование сообщений для Telegram по паттерну
4. Обработка ошибок и повторные попытки
5. Визуальный мониторинг выполнения workflow'ов

Роль в системе: "Руки" системы - автоматически выполняет рутинные задачи

ХОД РАБОТЫ

На рисунке 2 представлен скриншот запущенных контейнеров для работы ClearML, собрать его было непростой задачей, так как официальный билд у меня крашился, пришлось его подкорректировать для корректной работы.

Рисунок 2 – Скриншот контейнеров для ClearML

На рисунке 3 представлен UI ClearML раздела projects, перед этим надо было инициализировать среду выполнения через clearml-init.

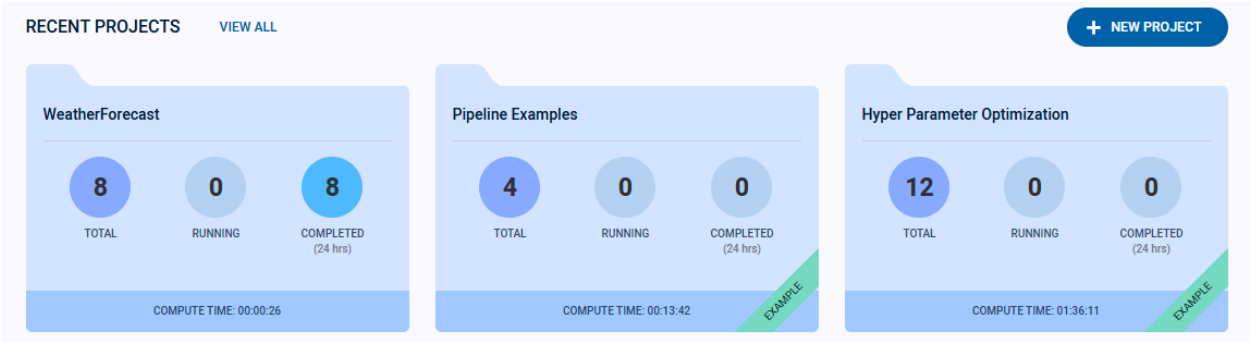

Рисунок 3 – Project ClearML

С помощью скрипта 01_create_dataset.py загружается датасет в ClearML. Датасет генерировался через апишку OPen Meteo, используя данные с 21 по 23 год для городов Austin,London, Tokyo и sydney. На рисунке 4 представлен загруженный датасет.

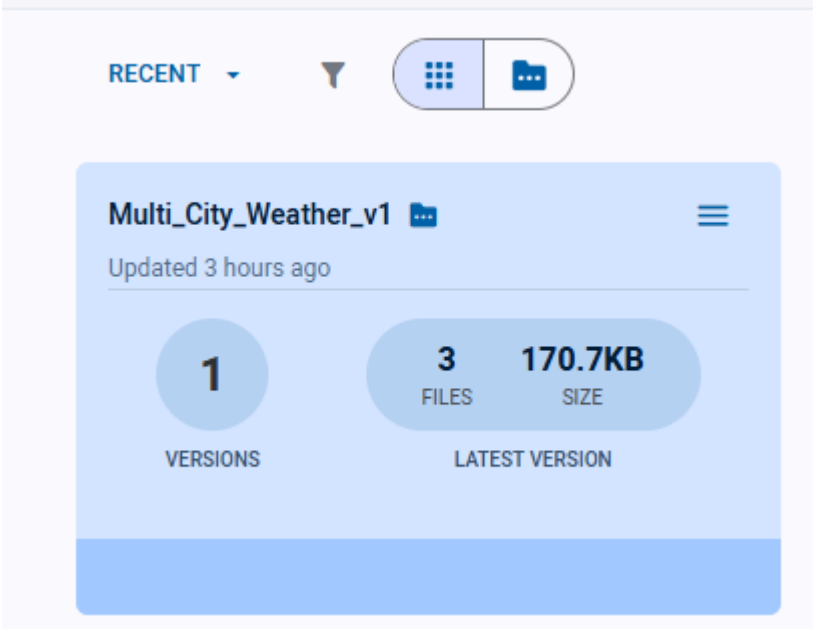

Рисунок 4 – Датасет в ClearML.

Далее был запущен скрипт 02_train_model.py. А после был этап HPO, чтобы затюнить модель.

Рисунок 5 – UI ClearML после hpo

После этого лучшую модель регистрируем по ID в ClearML.

```
Модель зарегистрирована в Model Registry: 65e30b5beafb4967b58839838817e97d
```

Рисунок 6 - Логи регистрации

FAST API + n8n часть

После обучения модели был поднят сервер для выполнения предикта, по тз /predict должен получать город и список дат, и возвращать прогнозы. Эндпоинт /predict реализован как гибридная система, где модель машинного обучения дополняется актуальными метеоданными через Open-Meteo API. Ключевая особенность - обогащение прогноза реальными историческими температурами. При получении запроса с городом и списком дат, сервис сначала обращается к Open-Meteo API за последними 30 днями исторических температур для указанного города. Эти реальные данные используются для заполнения лаговых признаков (Temperature_C_lag_1d, Temperature_C_lag_2d и т.д.) вместо синтетических значений.

В n8n все сильно проще чем во второй лабораторной работе. Здесь был поставлен shedule trigger который каждую неделю запускает скрипт по прогнозу погоды. Сначала генерируя даты для проноза - а потом делая запрос к развернутому серверу. После получения ответа - редактирует сообщение по

шаблону и отправляет в указанный чат.

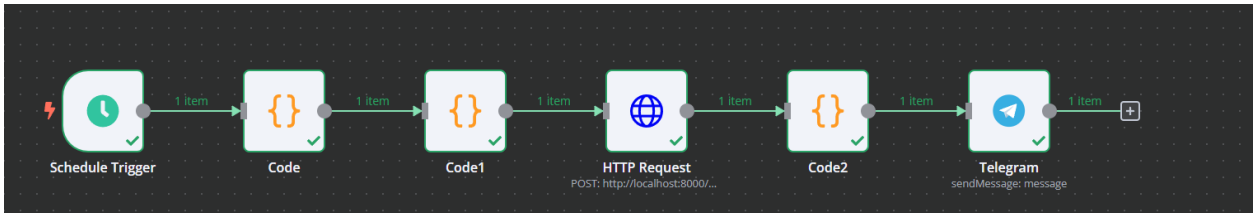

Рисунок 7 - Логи регистрации

Отправляется сообщение представленное на рисунке 8.

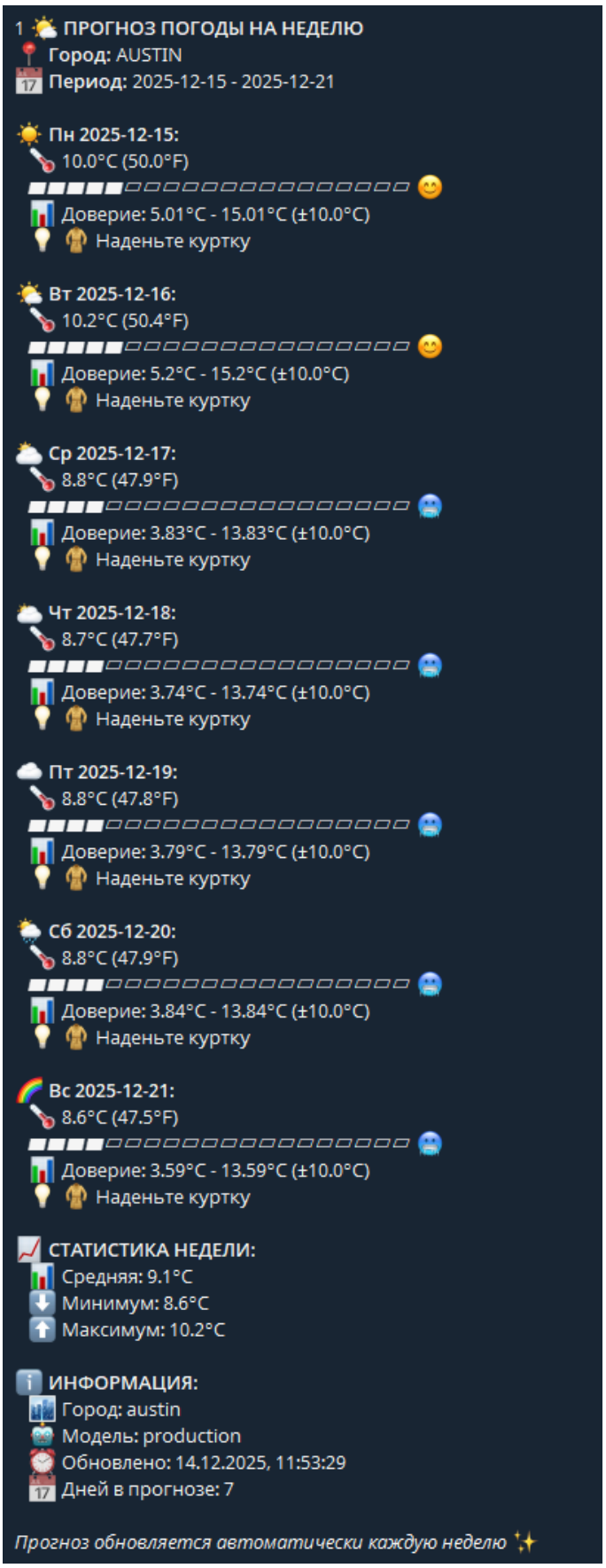

Рисунок 8 - Сообщение от бота

Общие моменты:

Система обладает высокой воспроизводимостью благодаря нескольким ключевым мерам. Во-первых, это достигается за счёт контейнеризации: все зависимости, включая версии библиотек Python, изолированы в Docker-образе. Во-вторых, используется фиксированный набор внешних источников данных — публичный API Open-Meteo, который гарантирует стабильный и документированный формат ответов. Таким образом, запуск системы в любой среде с использованием предоставленного Docker-образа приведёт к одинаковым результатам.

Обработка ошибок в системе реализована многоуровневая и охватывает весь процесс работы. На уровне внешних запросов она включает проверку доступности API Open-Meteo с использованием http и обработку статус-кодов. На уровне данных проводится валидация входных параметров и проверка структуры ответов от API. В случае сбоя пайплайн не прерывается полностью — для каждого города обработка происходит независимо, а ошибки логируются.Это гарантирует, что API остается доступным даже при частичных отказах.

Устойчивость системы дополнительно повышается за счёт логирования и наличия fallback-механизмов. Все операции, ошибки и важные события детально записываются в логи, что упрощает диагностику. Критические функции, такие как создание признаков для прогноза, имеют резервные сценарии. Вся обработка исключений централизована через глобальные обработчики FastAPI, которые гарантируют, что любой неожиданный сбой не приведёт к падению всего приложения, а вернёт пользователю структурированное сообщение об ошибке.

Файл основного докера в корне проекта - docker-compose.yml

файл для запуска inference внутри папки inference docker-compose.api.yml вместе с Dockerfile

библиотеки описаны в файл requirements.txt для общего докера и для inference

workflow.json - описание потока из n8n лежит в корне проекта

ВЫВОДЫ

Самый тяжелый этап лабораторной собрать докеры и запустить без ошибок….. С этим возникли сложности, так как в один момент я уже начал “жонглировать” версиями mongo и redis, чтобы у меня не крашился api и web server. В конечном итоге все запустилось, но с трудом. Подключение к ClearMl и все что с ним связанное стало уже не такой сложной задачей.

В ходе лабораторной работы был реализован полный ML-пайплайн в ClearML: от создания датасета и запуска экспериментов с подбором гиперпараметров до регистрации лучшей модели. Была обучена модель для предсказания погоды и добавлена в инференс-сервис на FastAPI в Docker. Также был настроен n8n-пайплайн, который раз в неделю автоматически запрашивает прогноз и присылает его в Telegram.