Домашка №4

- В данной домашке мы потренируемся в написании сервингов ML моделей и их нагрузочном тестировании
- Для обученной модели необходимо сделать REST сервис, у которого будет эндпоинт `/predict`, принимающий сырые входные данные для модели (вектор фич, текст, изображение, etc.), на выходе отдающий предсказание модели (предсказанный класс, число, текст, изображение, etc.)
- Сервис должен быть запускаться **в контейнере**, это может быть простой сервис на питоне (e.g. на fastapi), развернутый Triton Inference Sever или что-то на ваш выбор
- Для полученного сервиса необходимо провести нагрузочной тестирование. Для этого реализуйте скрипт, который в N параллельных соединений (N=1,2,5,10,20,50) отправляет некоторое множество запросов (e.g. 500), сохраняет время ответа и в конце выводит таблицу вида (N: avg, q25, q50, q90, q95, q99), т.е. распределение времени инференса модели. Здесь avg – среднее время выполнения, qXX – XX квантиль
- Модель необходимо приложить в гитхабе, либо в облаке. Результаты нагрузочного теста необходимо сохранить в ридми проекта, так же укажите характеристики железа (lscpu для CPU инференса, nvidia-smi для GPU).
- Важно! Следуйте рекомендациям вашего шаблона проекта, за нарушение структуры проверяющий оставляет за собой право снизить баллы
- **Важно!** В домашней работе предусмотрите, чтобы запуск экспериментов проводился в **контейнере**. При этом хорошей практикой является для каждого эксперимента запускать отдельных контейнер. В домашней работе это необязательно, однако за реализацию запуска набора экспериментов, где отдельный запуск происходит в своем контейнере, можно получить дополнительно +5 баллов;
- **Важно 2!** Если вы решили использовать W&B, то при сдаче домашки приложите ссылки на ваши эксперименты
- **Важно 3! Постарайтесь сделать ваш код модульным** и изолировать различные шаги (например, отдельно скрипт скачивания и отправки данных в S3 и отдельно скрипт обработки);
- **Важно 4! Итоговый проект с экспериментами и всеми необходимыми скриптами из прошлых домашек должен проходить настроенный в проекте линтер и тайпчекер!**

Алгоритм проверки домашки:

- Проверяющий клонирует репозиторий с проектом и разворачивает окружение по инструкции из README.md (или с помощью запуска bash/Makefile сценария). (возможно здесь стоит перейти на docker-compose для локального s3 и трекера экспериментов)
- Проверяющий убеждается, что все контейнеры поднялись и работают корректно;
- Проверяющий запускает скрипт нагрузочного тестирования и убеждается в его корректности

Важные уточнения:

- Итогом вашей работы всегда должен быть коммит в мастере через механизм PR, который прошел все проверки линтеров.

- Каждая лаба – логическое продолжение предыдущей, все изменения должны подливаться в осноной репозиторий в мастер.
- Код стоит поддерживать под unix (macos, linux), адаптировать под windows не нужно
- Шаблон, который собирается в первой лабе призван унифицировать работу с проектом. Следуйте тем правилам, которые в нем описаны.

Полезные советы:

- Постарайтесь аргументы, необходимые для запуска скриптов выносить в cli-аргументы (через argparse, click в питоне или через аргументы в bash); Для запуска скрипта **разработчик не должен лезть в код и хардкодить переменные**.
- Для создания конфигов с гиперпараметрами рекомендуется использовать **json**, либо **yaml**. Отдельно при использовании yaml-конфига, рекомендую посмотреть библиотеку OmegaConf;
- Рекомендую потренироваться и выполнить загрузку и выгрузку данных из S3 при помощи bash скриптов. (но это только рекомендация! На деле можно использовать питон или ваш любимый ЯП);