Домашняя работа MLsysd

НГУ, осень 2025

Общая информация

В качестве практической части курса вам нужно будет оформить одну задачу машинного обучения в репозиторий со всеми необходимыми компонентами. Смысл курса состоит в том, чтобы вы были способны самостоятельно произвести все необходимые операции и получить конечный результат, а не просто цифры и графики в jupyter notebook-е.
Мы будем разбирать устройство проекта для обучения нейронных сетей т.к. для них сейчас требуются наиболее продвинутые пайплайны.
Вам нужно будет описать выбранную задачу, создать репозиторий и работать над ним постепенно, выполняя выданные задания с помощью коммитов в основную ветку этого репозитория.
Ссылку на репозиторий вы передадите однажды, а далее я по ней буду проверять наличие сделанных заданий.

Задания на курс сразу выложены в этом файле, но приступать к ним вы сможете после прохождения соответствующих тем(либо заранее, если что то уже знаете и умеете). Проверка работ будет происходить по мере готовности работ по заданиям(заданий всего 2).

Крайний срок сдачи работ - за 2 дня до даты экзамена. Работы, присланные позже, будут учитываться только на пересдаче. Это, в частности, означает, что если в семестре ничего не делать, а прийти только на экзамен (даже со сделанной работой), то можно получить только баллы за экзамен и по итогу сдачи отправиться на пересдачу.

Экзамен будет проходить устно с опорой на проект (будут задаваться вопросы по нему, почему делали так, а не иначе, какие могут быть варианты), по формату это обычный устный экзамен, программу опубликуем ближе к концу когда пройдём большую часть.

Также в начале каждой лекции будет небольшой тест с простыми вопросами о предыдущем занятии. Они помогут вам вспомнить, что было в прошлый раз и настроиться на продолжение разговора. Кроме того это стимул чтобы смотреть лекции вовремя. Результаты этих тестов будут учтены в финальной оценке.

Составные части оценки:
- 30% - оценка проекта по критериям
- 60% - экзамен
- 10% - тесты по теории

Границы оценок:

- 85% - отлично
- 70% - хорошо
- 55% - удовлетворительно

Task 1: Project proposal

Вам необходимо выбрать тему проекта для дальнейшей реализации.
Описать умеренно детально, что вы собираетесь делать.
Выбор проекта будет обсуждаться на первых занятиях.

Шаблон описания проекта
Как работать с шаблоном:

- Заполнить название проекта и своё ФИ
    - название должно быть человекочитаемым и понятным неспециалисту в этой области
- Нужно сохранить структуру заголовков
- Внутри каждого заголовка вместо текста в кавычках нужно написать ваш текст для этого раздела. Необходимо именно ваше описание, просто оставить ссылку не будет достаточно.
- Во всех разделах нужно приводить ссылки на те или иные утверждения, компоненты, библиотеки, которые вы используете.

Требования к проекту работы:

- Над проектом можео работать командой до 5 человек
- у каждой группы должен быть уникальный проект
    - Данных и моделей сейчас чрезвычайно много
- Оформить проект нужно в виде документа .pdf на основе шаблона
- Использовать современные актуальные библиотеки машинного обучения
    - Не sklearn
    - Рекомендуем pytorch
- Сейчас используем только нейросети потому что они используют все обсуждаемые в курсе инструменты
    - Если вы очень хотите взять что-то иное, можно обсудить это со мной
- Данные должны быть реальными и не слишком маленькими
    - Репозиторий UCI не подойдёт
    - Синтетические данные не подойдут
    - Не слишком большие или сложные
- Нельзя брать уже оформленные репозитории(так как суть в том что вам самим его нужно оформить)
    - репозитории прошлых лет
    - репозитории от победителей соревнований на kaggle
    - вместо этого, например, возьмите за основу решение, оформленное в jupyter notebook, их очень много

В качестве источников вдохновения для выбора темы можем посоветовать:

- Вашу дипломную работу

- Соревнования на kaggle.com
    - Работы победителей
- Поиск Гугла по датасетам datasetsearch.research.google.com

Примеры задач для проектов:
- https://www.kaggle.com/competitions/learning-agency-lab-automated-essay-scoring-2/discussion/517014
- https://www.kaggle.com/competitions/cidaut-ai-fake-scene-classification-2024/overview

Если проект не будет соответствовать критериям, его нужно будет доработать согласно замечаниям потому что далее над ним нужно будет работать до конца курса. Надеемся, что таковых будут единицы.

Дедлайн выбора проекта: 15 октября

Task 2: Training code

Необходимо создать открытый репозиторий на github, в нём создать пакет на Python, который будет являться валидным с точки зрения пакетирования и решать вашу задачу (из предыдущей части).

В репозитории должны быть реализованы (детали ниже в разделах):
- загрузка данных
- препроцессинг данных (если необходимо)
- тренировка
- подготовка к продакшену
- инференс

Как мы будем проверять вашу работу (обобщённые шаги):
1. клонируем репозиторий
2. создаём новый чистый virtualenv (по инструкции)
3. устанавливаем зависимости (по инструкции)
4. `pre-commit install` - ожидаем успешную установку
5. `pre-commit run -a` - ожидаем успешный результат
6. запускаем тренировку
7. запускаем систему предсказания

Ниже описаны обязательные части вашего проекта: файлы и описание требований к основным этапам работы.

Список необходимых условий

Для получения оценки нужно выполнить все из них.
- репозиторий доступен по предоставленной ссылке
- сдаваемая версия кода находится в основной ветке репозитория (master или main). Остальные ветки не учитываются.

- тема проекта соответствует заявленной в первом задании
- реализованы (хотя бы в базовом виде) все основные разделы задания (помечены звёздочкой в заголовках)

README.md *

Фактически это инструкция по включению в проект вашего нового члена команды. Для этого вам понадобится описать две части:

- смысловое содержание проекта
- технические детали работы с ним

Первая часть это описание вашего проекта из предыдущего задания. Базово нужно его скопировать оттуда. Необходимо сохранить корректное визуальное представление (деление на заголовки разного уровня, структуру и проч). Возможно стоит поправить какие-то детали на основе того, что вы уже узнали.

Вторая часть касается технической стороны дела:

Setup

Должен присутствовать раздел Setup, где описывается процедура настройки окружения вашего проекта. Мы обсуждали множество различных вариантов (poetry, conda, uv и множество других развилок). Ваша инструкция должна привести к возможности продолжить разработку, а также запустить обучение и предсказание вашей модели. То есть должны быть настроены все необходимые инструменты.

Train

Должен присутствовать раздел Train, в котором рассказано, как запустить тренировку вашей модели. Если у вас есть несколько этапов (загрузка данных, preprocessing, несколько вариантов модели и проч), нужно описать каждый из них. Обязательно привести команды, которыми нужно запускать то или иное действие потому как мы обсуждали разные варианты работы с CLI)

Production preparation

Опишите шаги подготовки натренированной модели к работе, что для этого нужно сделать. Сюда могут входить перевод в onnx, tensorrt, etc.
Также в этом разделе можно описать комплектацию поставки вашей модели (какие артефакты, модули нужны для запуска).

Infer

Смысл тот же, что и у Train, но тут должно быть описано, как после тренировки запустить модель на новых данных. Также нужно описать формат таких данных, дать пример (можно в виде артефакта в вашем data storage).
Код предсказания должен зависеть от минимального количества зависимостей, поэтому его скорее всего не стоит реализовывать в одном файле с Train процедурой.

Dependencies*

Зависимости должны быть под управлением poetry или uv (на выбор), представлены в pyproject.toml, также не забудьте добавить poetry.lock или lock файл uv.
Зависимости должны успешно устанавливаться, с ними должен успешно запускаться код проекта.
Не должно быть лишних (не используемых в проекте) зависимостей.

Code quality tools*

Необходимо использовать pre-commit с базовыми хуками pre-commit, black, isort, flake8, prettier (для не Python файлов).
Все хуки должны быть соответствующе настроены.
Также необходимо, чтобы запуск `pre-commit run -a` не выдавал ошибок (зелёные результаты).

Training framework*

Базовой опцией является PyTorch Lightning, также вы можете воспользоваться другими фреймворками, которые мы обсуждали, но в них меньше возможностей.
Обучение должно быть сделано с помощью выбранного фреймворка, используя доступные в нём возможности (не нужно строить велосипеды).

Data management *

Необходимо использовать dvc для хранения данных.
В качестве хранилища можно использовать гугл диск, либо s3 (убедитесь, что оно доступно), либо локальное хранилище.

Скачивание данных с помощью dvc необходимо встроить в имеющиеся команды train и infer, для этого у dvc есть python api (на крайний случай можно использовать CLI из Python кода).

Если вы используете локальное хранилище, необходимо написать функцию `download_data()`, которая скачивает ваши данные из открытых источников.

Hydra*

Переведите основные гиперпараметры препроцессинга, обучения и постпроцессинга в yaml конфиги hydra. Сами конфиги лучше всего расположить в папке configs в корне репозитория.
Конфиги должны быть сгруппированы по тем операциям, которые вы проводите (например свой файл для препроцессинга, другой для основной модели, третий для сервинга и т.д.). Используйте иерархические конфиги.

В коде не должно быть магических констант. Их можно вынести либо в отдельный Python файл (если они глобальные и их не предполагается менять), либо под управление hydra.

Logging *

Необходимо добавить логирование ваших основных метрик и функций потерь (всего не менее 3 графиков). Также в эксперимент записывать использованные гиперпараметры и версию кода (git commit id). Считайте, что сервер mlflow уже поднят по адресу `127.0.0.1:8080` (вы можете поднимать его локально для тестов по этому же адресу). Адрес добавьте в поле конфига.

Графики и логи положите в отдельную директорию в репозитории с названием plots.

Можно использовать дополнительные системы логирования (например wandb), если вам это нужно.

Model production packaging

Переведите вашу модель в onnx (5 баллов). Также можно перевести пайплайны препроцессинга и постпроцессинга (если они не тривиальны в вашем проекте).
Эту часть можно сделать прямо в конце пайплайна обучения, либо как отдельную команду, тогда опишите это в README.

Переведите вашу модель в TensorRT (5 баллов). Оптимально это сделать из готового файла onnx, либо можно непосредственно из модели. Оформите такой перевод в виде shell файла или python cli команды. Если вы используете пример данных для такого перевода, их тоже нужно добавить в dvc.

Inference server

Можно реализовать двумя подходами: с помощью MLflow Serving (макс 5 баллов) или Trion Inference Server (макс 10 баллов).
В разделе Infer README нужно описать, как его поднять и использовать.

Список типичных ошибок

Что нужно делать

- Не должно быть исполняемого кода на уровне файла. Используйте в таких случаях `if __name__ == '__main__':`
    - В частности нельзя объявлять переменные (кроме констант) на верхнем уровне файла (не внутри функции или класса)
- Не использовать `warnings.filterwarnings("ignore")`. Никогда не делайте этого в ~~продакшен~~ любых проектах. Это огромный задел на отстреливание себе ноги. Люди пишут предостережения чтобы вас предостеречь - сделайте необходимые для этого действия

- Нельзя сохранять данные в гит!!!!!!!!!!!!! То есть файлы .json, .csv, .h5 и проч. То же касается файлов натренированных моделей (.cbm, .pth, .xyz, .onnx, etc).
- Название репозитория (в т.ч. url) должно отражать смысл вашего проекта (e.g. `cats-vs-dogs`), а не название курса (e.g. `mlops_homework`). Использовать - [dash] в качестве разделителя (не _ [underscore])
- Нужно назвать питон пакет (aka папка с вашим кодом) по правилам питона (`snake_case`), а не любым другим способом (e.g. `MYopsTools`)
- Также питон пакет необходимо называть согласно смыслу вашего проекта, не `src` или `my_project`
- Используйте дефолтный .gitignore для Питона (а не пустой), его дополняйте необходимыми вам путями. Дефолтный конфиг гуглится и даже предлагается вам гитхабом при создании репозитория.
- Файлы с кодом должны называться в `snake_case`, не в `CamelCase` (e.g. `Dataset.py`)
- Не использовать argparse, вместо него fire, click, hydra или другой инструмент для cli.
- Не использовать однобуквенные переменные кроме `i, j, k`. Вместо этого давайте переменным семантически наполненные имена (`data` или `features` вместо `X` и т.д.)

Что делать желательно

- Под вызовом `if __name__ == '__main__':` лучше вызывать ровно одну функцию (можно её назвать main или как-то ещё), а не писать всю логику непосредственно под if-ом.
- Использовать fire совместно с hydra через compose api
- Сделать одну входную точку commands.py, где вызывать соответствующие функции из файлов
- Используйте `pathlib` вместо `os.path` - важа жизнь заиграет совершенно другими красками

Список ошибок не является полным. Способов сделать странные вещи очень много, поэтому старайтесь применять знания, полученные в курсе, а также ваше чувство прекрасного.

P.S. если что-то непонятно или вы хотите сделать как-то иначе - пишите в чат, обсудим. Данные правила не являются догмами (хотя в этом ДЗ необходимо их выполнить), а скорее набором хороших техник, которые, впрочем, нужно адаптировать под задачу и ситуацию в целом.