DataFest 2020 Notes

Ilnur Shugaepov
VK.com

September 2020

Оглавление

ML REPA

Трек посвященный воспроизводимости в ML. Много разговоров о том

- В чем схожесть и отличия обычного процесса разработки ПО и процесса разработки ML моделей
- Какие инструменты и подходы, используемые при классической разработке ПО, можно переиспользовать при разработке ML продуктов
- Каким образом можно добиться воспроизводимости

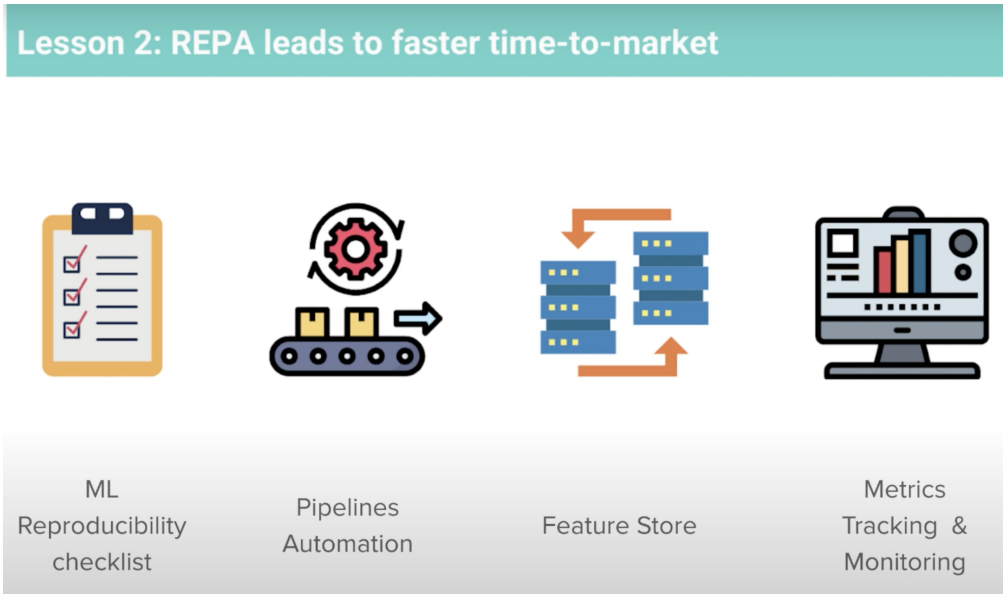
Remark 1. *Наши рекомендации.*

1.1 Machine Learning REPA in Action

Video: `https://youtu.be/sZTO6LihuAM`

Доклад про lessons learned и good practices, которые позволяют сделать процесс разработки ML моделей воспроизводимым.

Time-to-market время при разработке новой модели можно сильно уменьшить если есть следующие ключевые компоненты:



1.1.1 ML Reproducibility checklist

1. Environment dependencies control — нужно всегда трекать зависимости (версии баблиотек итд)
2. Code version control — использовать git для разработки (код ревью экспериментов)
3. Control run params
4. Automated pipelines — собирать пайплайны для обеспечения воспроизводимости
5. Artifacts version control — например, трекать версии моделей которые получаются на выходе
6. Experiments results tracking — сохранять результаты экспериментов (метрики/артифакты)
7. Automated CI/CD — авто-тесты/автоматическая выкатка в прод

1.1.2 Good Practices

Project Organization Во всех ML проектах команды должна быть похожая структура - это позволяет проще кооперироваться.
В качестве примера структуры ML проекта приводят Cookiecutter Data Science[1].

Task Tracking Для любого изменения в коде должна быть соотвествующая задачка в таск-трекере.

Documentation Как минимум в папках с проектом должны быть README.md файлы, в которых кратко описаны основные моменты.
Не стоит забывать и про документацию в confluence.

Testing Нельзя пренебрегать тестированием (см. Раздел 1.5).

1.2 [MTS] Software engineering life hacks for Data Science

Video: https://youtu.be/i-SrzrzpieI

В данном докладе говорилось по большому счету о том же, что есть в наших рекомендациях к воспроизводимости:

- Упорядоченный репозиторий
- Чистый код (прямая отсылка к книжке)
- Рефакторинг
- Работа в отдельных ветках и обязательное код ревью.
  Во время ревью следует особое внимание уделить следующим аспектам
  - DS ошибки
  - Воспроизводимость
  - Ошибки разработки
- Полезные расширения для Jupyter
  - Extensions[2] - Execute time, Autopep8, Table of Contents, Collapsible Headings

[1] https://drivendata.github.io/cookiecutter-data-science/
[2] https://github.com/ipython-contrib/jupyter_contrib_nbextensions

- Papermil[3] - инструмент для параметризации и удобного запуска ноутбуков. Позволяет удобно запустить ноутбук с заданными параметрами.

1.3 [OK.ru] Reproducibility is not a pain anymore!

Video: https://youtu.be/arTk1YvgRls

ML модель = Данные + Код + Зависимости

По большому счету повторение предыдущего доклада Миши

Используют DVC для работы с hdfs файлами в том числе.

1.4 Comparison of frameworks for pipelines automation: Airflow vs. Prefect

Video: https://youtu.be/Qx09k-bmdBU

Prefect[4] - так же как и Airflow, является инструсментом для автоматизации пайплайнов. Был создан основными контрибьютерами Airflow.

Прямо из доклада не очень понятно, почему Prefect лучше чем Airflow, но на medium'е есть большой пост[5] от авторов Prefect про преимущества в сравнении с Airflow
Самое очевидное улучшение - более понятный и приятный UI.

[3] https://github.com/nteract/papermill
[4] https://github.com/PrefectHQ/prefect
[5] https://medium.com/the-prefect-blog/why-not-airflow-4cfa423299c4

1.5 [MTS] Testing for Data Science Hands-on Guide

Video: https://youtu.be/GgL4BKWYlx8
Code: https://gitlab.com/Julia_chan/testing-for-data-science

Что тестировать	Пример из DS
Functional tests: результат, а не реализация	Предобработка фичей
Non-functional tests: быстродействие	Модель должна отвечать меньше чем за 1с
Model performance: распределение предиктов	Распределение ответов не должно сильно меняться

1.5.1 Библиотеки и фрэймворки для тестирования

- Pytest[6] — удобный для DS проектов
- Great Expectations[7][8] — фрэймворк для тестирования данных
- coverage.py — для проверки покрытия тестами
- Hypothesis[9]

1.5.2 Отличия продакшн кода и тестов

- Тесты проще, чем вещи которые тестируем
- Тест должен быть понятен буз чтения кода, чтобы при ошибке было легко искать причину
- Рефакторинг кода делает тест проще
- Тестировать нужно поведение, а не реализацию

Когда писать тесты:

- Во время написания кода программы
- Перед рефакторингом
- Когда найдены ошибки в коде

1.6 [Raiffeisenbank] How I Stop Worrying and Love the Standartization

Video: https://youtu.be/I9c8UqaRL6M

Проблема команда растет, каждый делает свои модели как хочет, в результате никто не может поддерживать решения коллег по команде.

[6] https://docs.pytest.org/en/stable/
[7] https://github.com/great-expectations/great_expectations
[8] https://medium.com/@expectgreatdata/down-with-pipeline-debt-introducing-great-expectations-862ddc46782a
[9] https://hypothesis.readthedocs.io/en/latest/

Желания

- Унифицировать подход к разработке моделей
- Упрощение поддержки готовых моделей
- Сокращение времени на разработку
- Прозрачность и воспроизводимость результатов

Решение

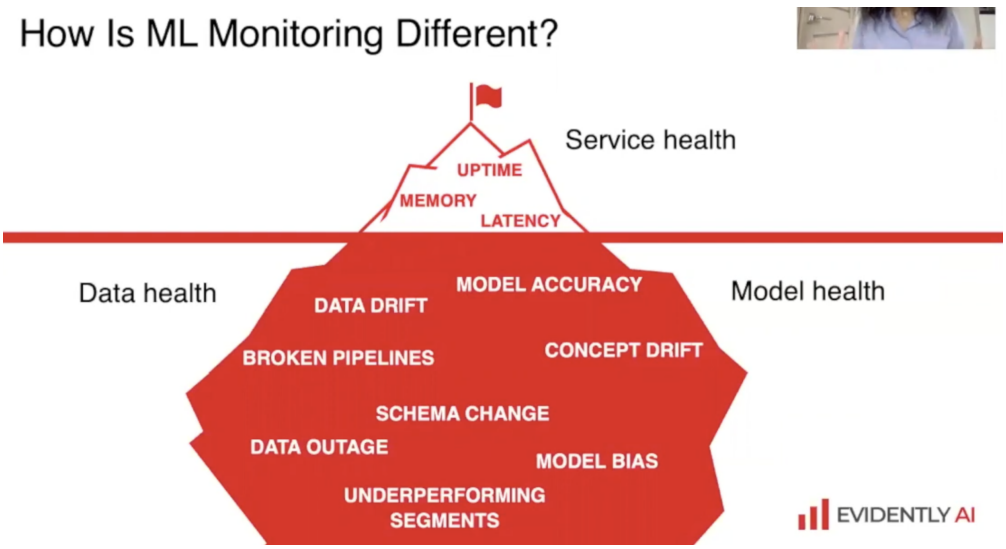
1. Своя библиотека для унифицированного сбора данных и обучения моделей (прям как `vk_ads`)
2. Автоматизация пайплайнов с помощью Airflow
3. Spark Tests — тестирование данных (проверка схем, проверка числа строк, проверка на неожиданные None)
4. Автоматическое заполнение confluence — после того как отработает airflow пайплайн, информация о метриках и пр. автоматически пишется в confluence

Управление DS проектами

Использование Kanban в DataScience команде:
https://drive.google.com/file/d/1fXJlM_sdSQTc8kXI8aMIftTCHZj2_AhG/view.
(к сожалению только презентация без видео)

1.7 The day after deployment: how to set up your model monitoring

Video: https://youtu.be/2QnSSOFrP14



Remark 2. *В терминах рисунка 1.1 у нас маловато ML-focused reports.*

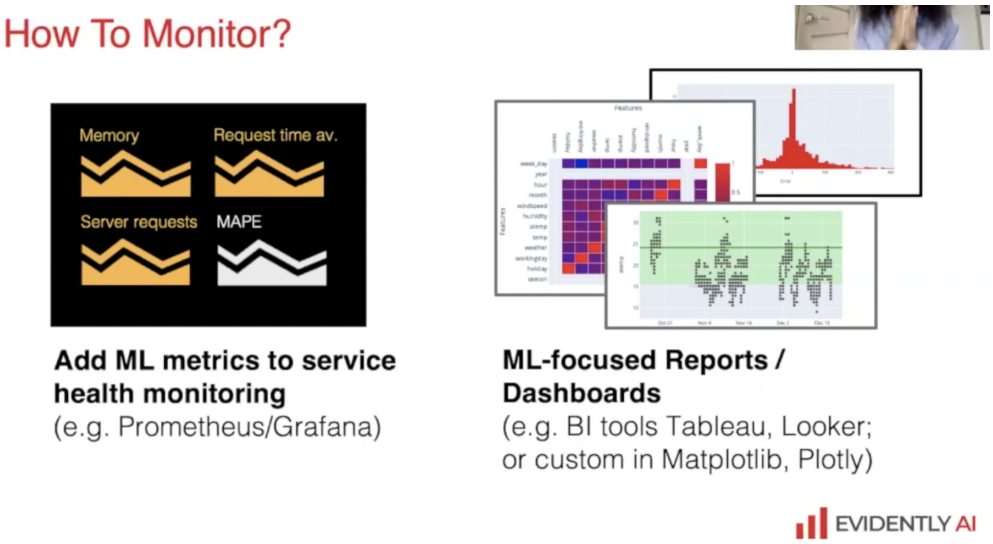


Рис. 1.1: How To Monitor?

Key Takeaways:

1. Продолжать использовать рекомендации (не забывать про code-review)
2. Подумать над тем, чтобы создать Feature Store (пример: то как организовано хранение признаков в хадупе Mail.ru)
3. Попробовать посмотреть на DVC + hdfs
4. Писать тесты (хотя бы для кода, который используется в продакшн пайплайнах)
5. Тестирование данных важно

Interpretable ML

К сожалению, на момент написания текста в записи был доступен только один доклад.

2.1 ML Interpretability Problems in Tabular Data Tasks

Video: https://youtu.be/j0fl9_utKx8

Основные рекомендации из первого доклада

- Perform error analysis — нужно найти примеры, на которых модель ошибается сильнее всего, и понять почему это происходит, например, посмотрев на SHAP values.

  Это анализ может помочь сгенерировать новые идеи для feature-engineering'а, чтобы улучшить модель.
- Start with simple models — при работе над новой задачей, всегда хорошо начинать с простых моделей, которые легко интерпретировать, например, с логистической регрессии, неглубоких решающих деревьев.
- Remove biases from data.
- State-of-the-art подход к анализу важности фичей - SHAP[1] values, но есть и другие интересные подходы, к которым стоит присмотреться: Lime[2], Permutation importance[3].

[1] https://github.com/slundberg/shap
[2] https://github.com/marcotcr/lime
[3] https://scikit-learn.org/stable/modules/permutation_importance.html

Graph ML

Remark 3. *На прошедшей конференции KDD 2020 30% всех статей было посвящяно Graph ML, так что это определенно область, к которой стоит присмотреться повнимательнее.*

Статья[1] про интересные приложения Graph ML, в том числе для построения рекомендаций.

3.1 [Google] Unsupervised Graph Representations

Video: https://youtu.be/J3h_15iu4gE

Обзорный доклад посвященный большому числу работ по обучению embedding'ов вершин графов. В докладе излошены основные идеи методов.

3.2 [Skoltech] Link Prediction with Graph Neural Networks

Video: https://youtu.be/WNQi_kvdlr4

Еще один обзорный доклад посвященный задаче предсказания ребер с помощью графовых нейросетей.

[1] https://towardsdatascience.com/top-trends-of-graph-machine-learning-in-2020-1194175351a3