

Автоматизация и версионирование ML – экспериментов. Почему это важно как легко этого добриться

Никита Варганов

29.04.2020

Варганов Никита



Выпускник МГТУ имени Н. Э. Баумана
kaggle competitions master, top 500
Senior Data Scientist в Сбербанке

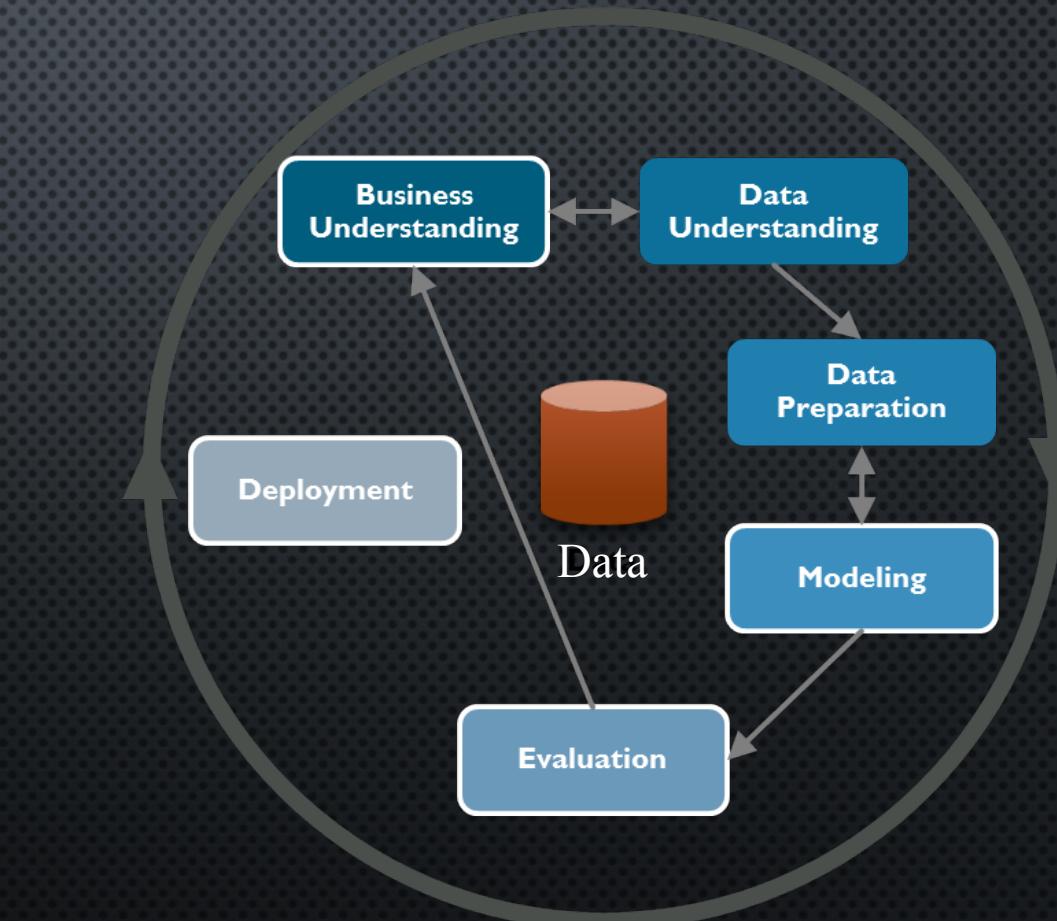
Содержание

1. Как выглядит типичный проект по Data Science
2. Как работает Data Scientist в 2020
3. Проблемы, с которыми сталкиваются Data Scientist'ы
4. Sacred – инструмент для организации DS – экспериментов
5. Пример создания рабочего проекта

Жизненный путь DS-проекта

CRISP DM – межотраслевой процесс для исследования данных

1. Выявление потребностей бизнеса
2. Предварительный анализ данных
3. Подготовка данных
4. Моделирование
5. Оценка результатов
6. Внедрение



Жизненный путь DS-проекта

- 1. Вывявление потребностей бизнеса.....→ Определение бизнес – целей
- 2. Предварительный анализ данных Определение ключевых участников
- 3. Подготовка данных Оценка текущей ситуации
- 4. Моделирование Подготовка плана проекта
- 5. Оценка результатов
- 6. Внедрение

Жизненный путь DS-проекта

1. Выявление потребностей бизнеса
 2. Предварительный анализ данных
 3. Подготовка данных
 4. Моделирование
 5. Оценка результатов
 6. Внедрение
- → Сбор данных
Описание данных
Исследование данных
Оценка качества данных

Жизненный путь DS-проекта

1. Выявление потребностей бизнеса
 2. Предварительный анализ данных
 3. Подготовка данных
 4. Моделирование
 5. Оценка результатов
 6. Внедрение
- →
- Отбор данных
 - Очистка данных
 - Генерация новых данных
 - Интеграция данных

Жизненный путь DS-проекта

1. Выявление потребностей бизнеса
 2. Предварительный анализ данных
 3. Подготовка данных
 4. Моделирование
 5. Оценка результатов
 6. Внедрение
-
- Выбор алгоритма
Планирование тестирования
Обучение модели
Оценка результатов

Жизненный путь DS-проекта

1. Выявление потребностей бизнеса
2. Предварительный анализ данных
3. Подготовка данных
4. Моделирование
5. Оценка результатов
6. Внедрение

Формулирование результатов в
терминах бизнеса
Анализ проверенных гипотез
Принятие решения

Жизненный путь DS-проекта

1. Выявление потребностей бизнеса
2. Предварительный анализ данных
3. Подготовка данных
4. Моделирование
5. Оценка результатов
6. Внедрение

Планирование развертывания
Настройка мониторинга модели
Составление отчета о работе модели

**50 – 80 % рабочего времени
Data Scientist тратит на
работу с данными!**

Какие метаданные необходимо сохранять?

1. Данные
2. Модель
3. Шаги подготовки данных
4. Гиперпараметры модели
5. Метрики
6. Контекст

Какие метаданные необходимо сохранять?

1. Данные
2. Модель
3. Шаги подготовки данных
4. Гиперпараметры модели
5. Метрики
6. Контекст

Набор данных для обучения
Набор данных для тестирования
Имена и типы переменных
Статистика по переменным
Распределение признаков
Распределение целевой переменной

Какие метаданные необходимо сохранять?

1. Данные
 2. Модель → Название и версия библиотеки
Название модуля, связанного с моделью
 3. Шаги подготовки данных
 4. Гиперпараметры модели
 5. Метрики
 6. Контекст
- Пример:** xgb.XGBClassifier и xgb.Booster

Какие метаданные необходимо сохранять?

1. Данные
2. Модель
3. Шаги подготовки данных.....→
 - Обработка категориальный признаков
 - Обработка пропущенных значений
 - Масштабирование / центрирование
 - Обработка выбросов
 - Объединение данных
 - Агрегация данных
4. Гиперпараметры модели
5. Метрики
6. Контекст

Какие метаданные необходимо сохранять?

1. Данные
2. Модель
3. Шаги подготовки данных
4. Гиперпараметры модели.....→
 - Гиперпараметры финальной модели
 - Гиперпараметры перебираемых моделей
 - Значения метрики качества при каждом значении гиперпараметров
5. Метрики
6. Контекст

Какие метаданные необходимо сохранять?

1. Данные
2. Модель
3. Шаги подготовки данных
4. Гиперпараметры модели
5. Метрики.....→
 - Значение метрики на обучающей выборке
 - Значение метрики на валидационной выборке
 - Значение метрики на кросс – валидации
6. Контекст

Какие метаданные необходимо сохранять?

- 1. Данные
- 2. Модель
- 3. Шаги подготовки данных
- 4. Гиперпараметры модели
- 5. Метрики
- 6. Контекст.....→
 - Исходный код
 - ЯП и его версия
 - Зависимости
 - Информация о машине
 - Системные пакеты

Какие метаданные необходимо сохранять?

- 1. Данные
- 2. Модель
- 3. Шаги подготовки данных
- 4. Гиперпараметры модели
- 5. Метрики
- 6. Контекст.....→
 - Исходный код
 - ЯП и его версия
 - Зависимости
 - Информация о машине
 - Системные пакеты

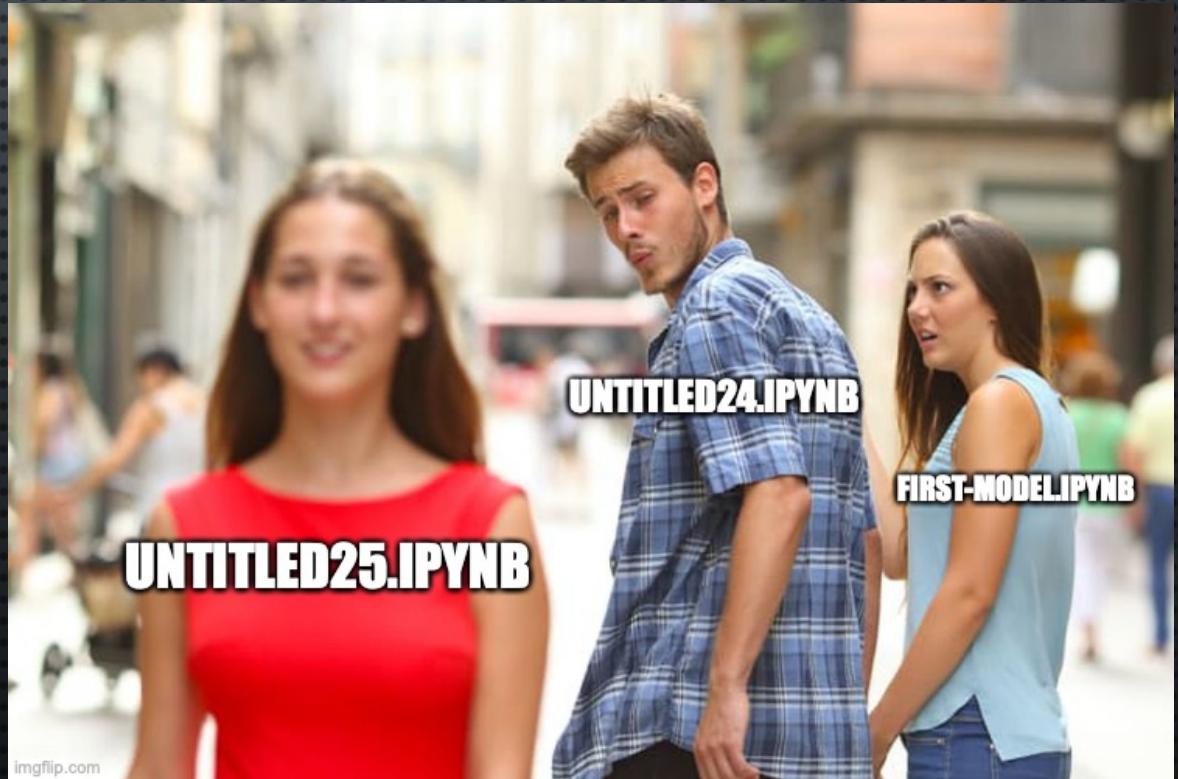
Инструменты типичного DS



Apache
Zeppelin

Почему Jupyter Notebook плох?

1. Практически невозможно управлять кодом / версиями кода
2. Сложно тестировать код
3. Нелинейный workflow



Проблемы типичного Data Scientist'a

1. Не использует системы контроля версий
2. Работает в произвольном python – окружении
3. Не использует IDE
4. Не использует наработки с предыдущих проектов из – за низкого качества кода / не универсальности решения
5. Отсутствует структура проекта
6. Отсутствует документация
7. Отсутствуют тесты





Любой код, который участвует в бизнес – процессе, должен рассматриваться как production код

Sacred

Sacred – это инструмент для настройки, организации, регистрации и воспроизведения вычислительных экспериментов.

Основной функционал:

1. отслеживание всех параметров вычислительного эксперимента;
2. легкий запуск экспериментов с разными настройками;
3. сохранение конфигурации запусков;
4. воспроизводимость результатов

Страница проекта: <https://sacred.readthedocs.io/en/stable/index.html>

Декораторы в python

Декоратор – это функция, которая принимает на вход другую функцию и расширяет ее функциональность без непосредственного изменения ее кода.

```
import time

def time_evaluation(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        res = func(*args, **kwargs)
        elapsed_time = time.time() - start_time
        print(f"func: {func.__name__}, elapsed_time: {elapsed_time}")
        return res
    return wrapper

@time_evaluation
def test_function(sleep_time: int):
    time.sleep(sleep_time)
    print(f"slipped: {sleep_time} sec.")
    return sleep_time

if __name__ == "__main__":
    res = test_function(5)
    assert res == 5
```

Experiment & Observers

Experiment – центральный класс Sacred, который отвечает за создание эксперимента.

Observers – класс для отслеживания и логирования мета-информации вычислительных экспериментов.

```
from sacred import Experiment  
from sacred.observers import FileStorageObserver
```

```
ex = Experiment(experiment_name)  
ex.observers.append(FileStorageObserver("runs"))
```

Декораторы sacred

1. `@ex.automain` – применяется к основной функции эксперимента; при запуске эксперимента – основная функция будет запущена на выполнение.
2. `@ex.config` – применяется к функции, в которой определена конфигурация эксперимента, выполняется перед началом эксперимента.
3. `@ex.capture` – применяется к функциям, которым требуется передать в качестве аргумента значение из конфигурации эксперимента.
4. `@data_ingredient` – применяется к функциям, которые могут быть использованы повторно в разных DS – проектах.

Ссылки

1. Материалы с вебинара – [GitHub](#)
2. <https://sacred.readthedocs.io/en/stable/>
3. <https://mlinproduction.com/ml-metadata-tutorial/>
4. <https://www.hhlcks.de/blog/2018/5/4/version-your-machine-learning-models-with-sacred>
5. I don't like jupyter notebooks – Joel Grus - [youtube](#)

Спасибо за внимание!

Вопросы?



@nv_27



@nv27



fb.com/nico.varganov