Промышленная эксплуатация моделей машинного обучения

SELEZNEV ARTEM
SENIOR DE @ SBER

DEPLOY ML

SELEZNEV ARTEM
SENIOR DE @ SBER

3…2…1…

tg: @SeleznevArtem

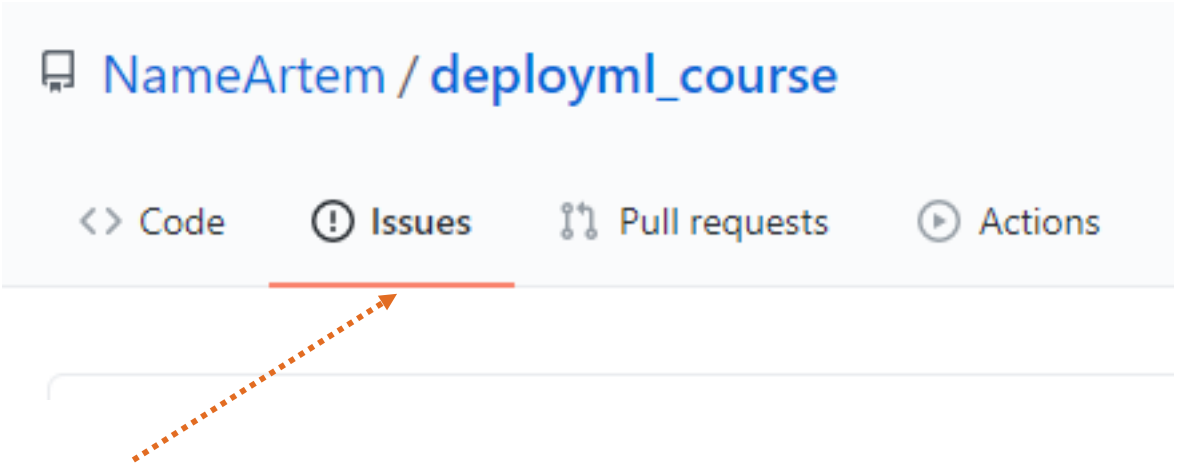
/NameArtem

/seleznev-artem

/seleznev.artem.info

https://github.com/NameArtem/deployml_course

https://github.com/NameArtem/deployml_course



ЧТО НЕОБХОДИМО

Инструменты автоматизации

Linux

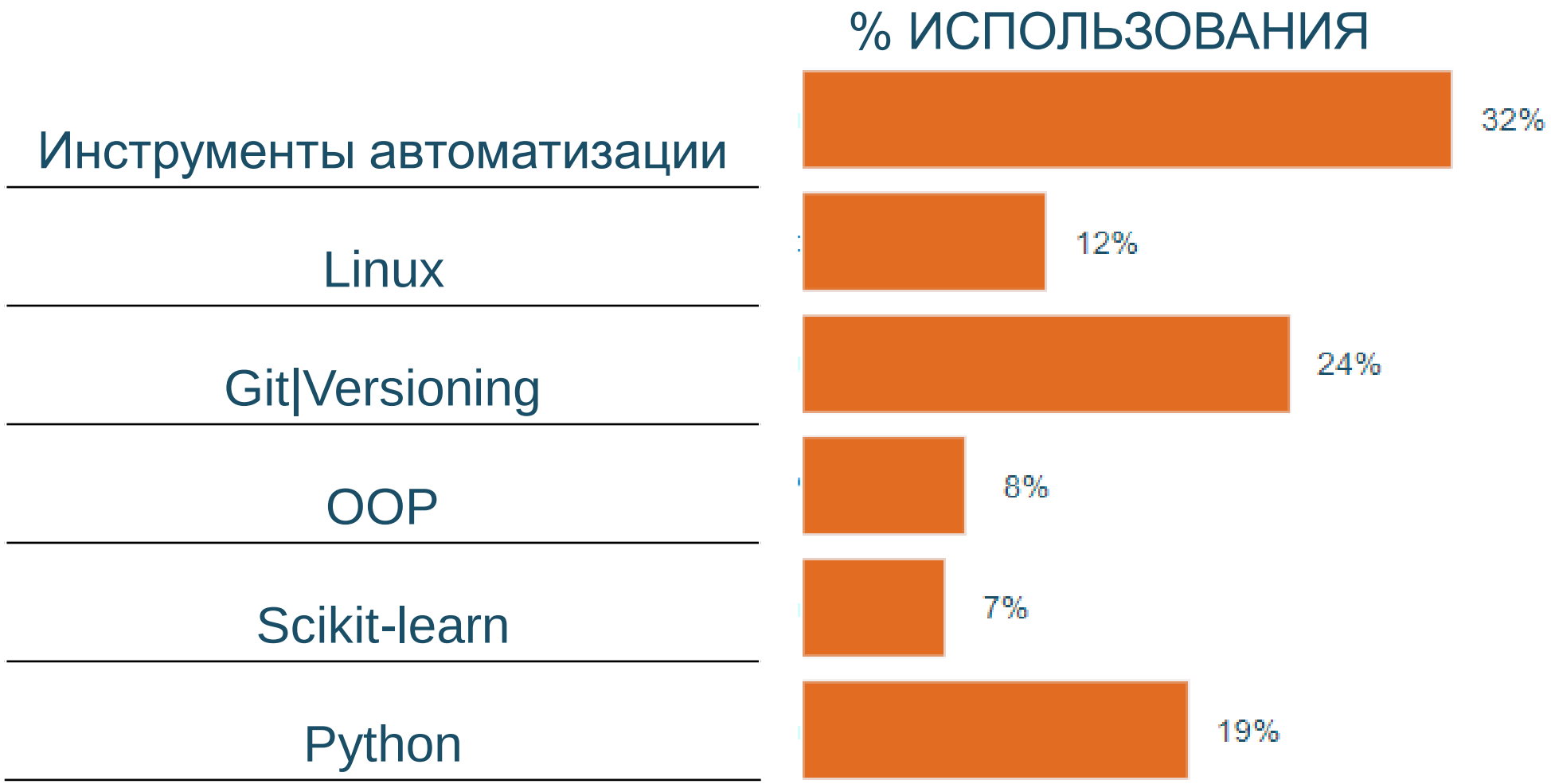
Git|Versioning

OOP

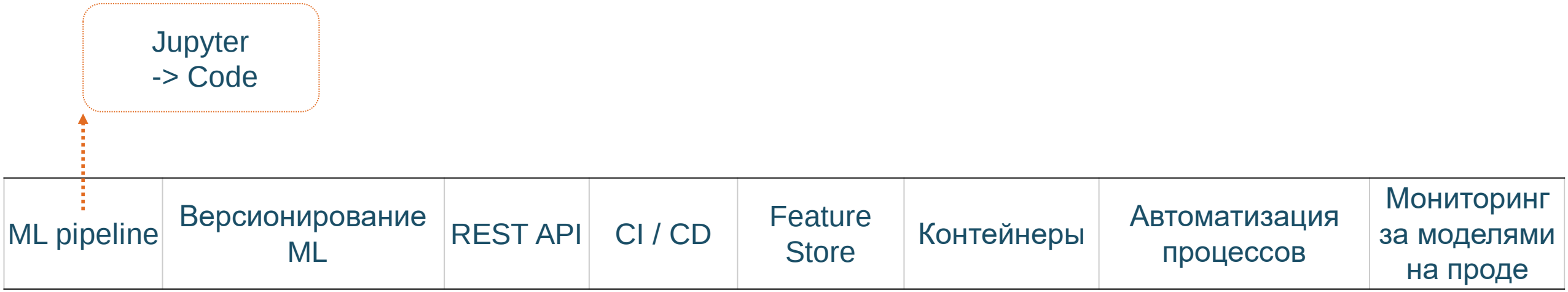
Scikit-learn

Python

ЧТО НЕОБХОДИМО



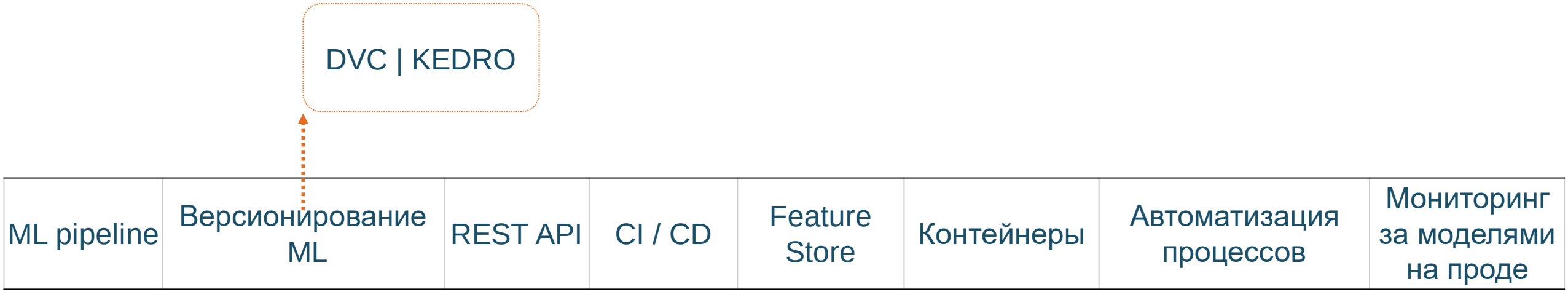
ОБЗОР КУРСА



\+ Преобразование Jupyter notebook исследования в ML проект (виды pipeline)

! Куда девать повторяющийся код и его правильно переиспользовать

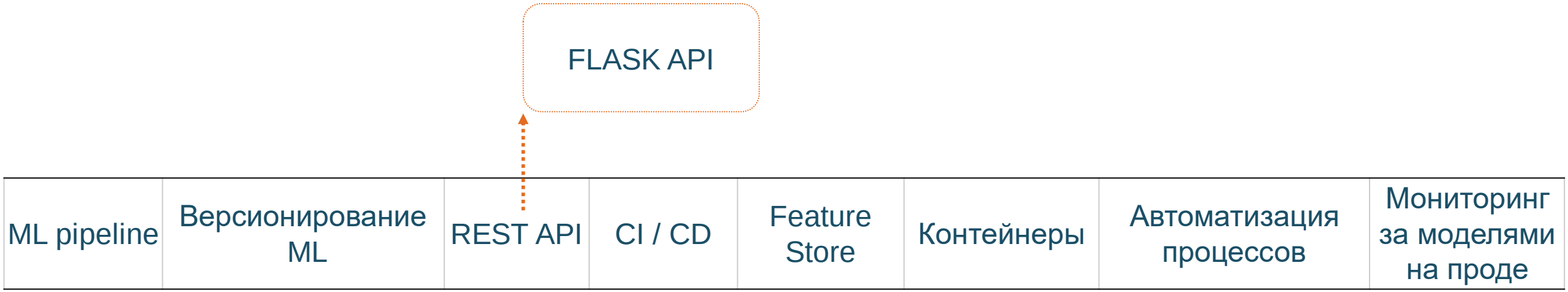
ОБЗОР КУРСА



+ Делаем репрезентативный процесс разработки (повторяем процесс у любого ДС)
+ Проекция практик разработчиков в процессы ML

! Нет процесса печальнее на свете, чем версионирование процесса разработки ДС.

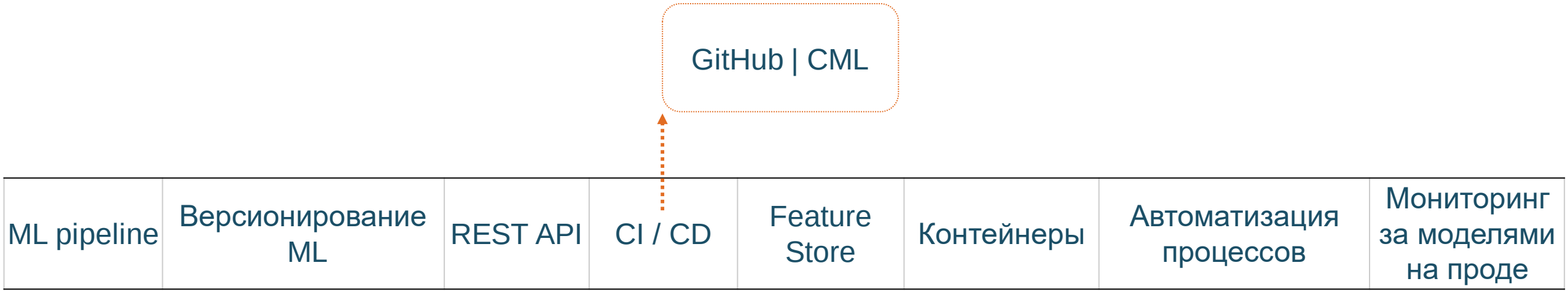
ОБЗОР КУРСА



+ Разрабатываем в архитектуре
  “learn by batch / predict on fly / serve with API”

! API не только для поставки данных / результатов

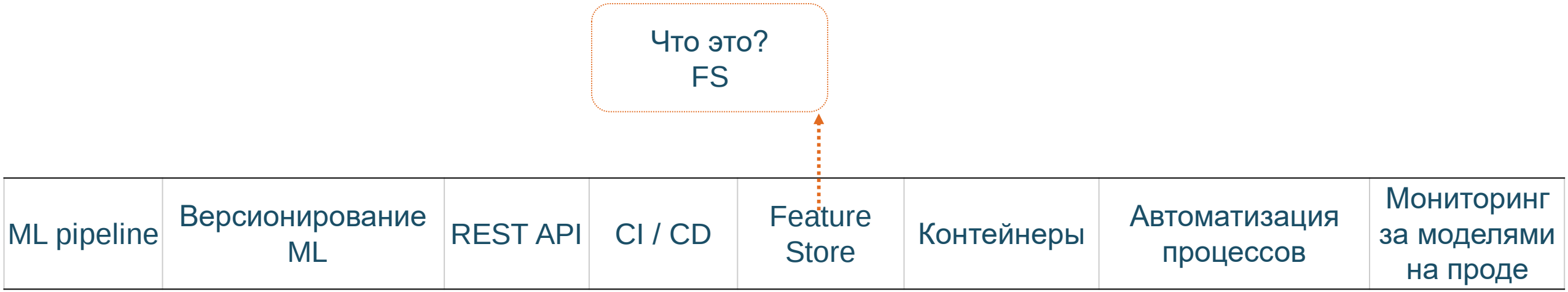
ОБЗОР КУРСА



+ Постигаем MLOps
+ Как сделать процесс постоянной разработки с учетом особенностей ML

! Разрабатываем тесты с которыми нельзя жульничать
assert True == True

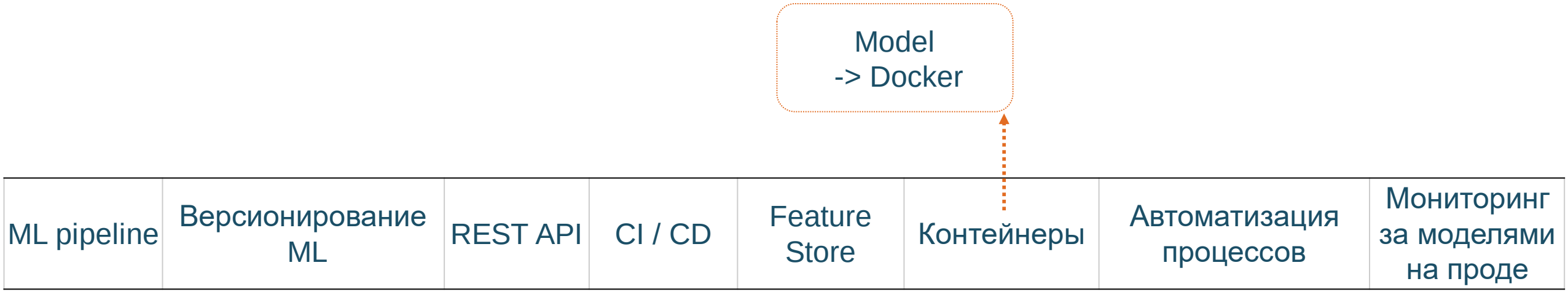
ОБЗОР КУРСА



\+ Инструмент для ускорения разработки (делимся «фичами» между ДС)
\+ Мониторинг за качеством данных

! Уменьшаем время выхода на рынок (TTM) моделей

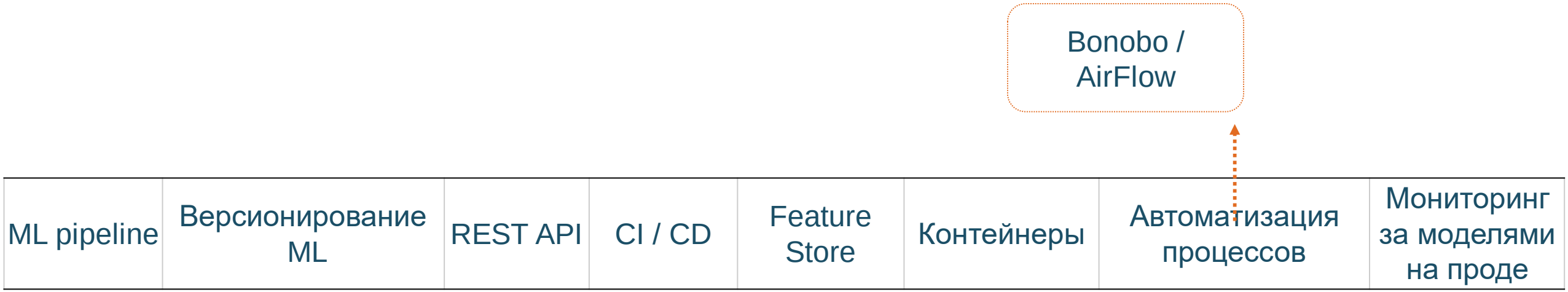
ОБЗОР КУРСА



+ Docker != VM на вашем компьютере
+ Жизненный цикл ML модели
  (до прода и обратно)

! ДС «песочница» не соответствует вашему проду

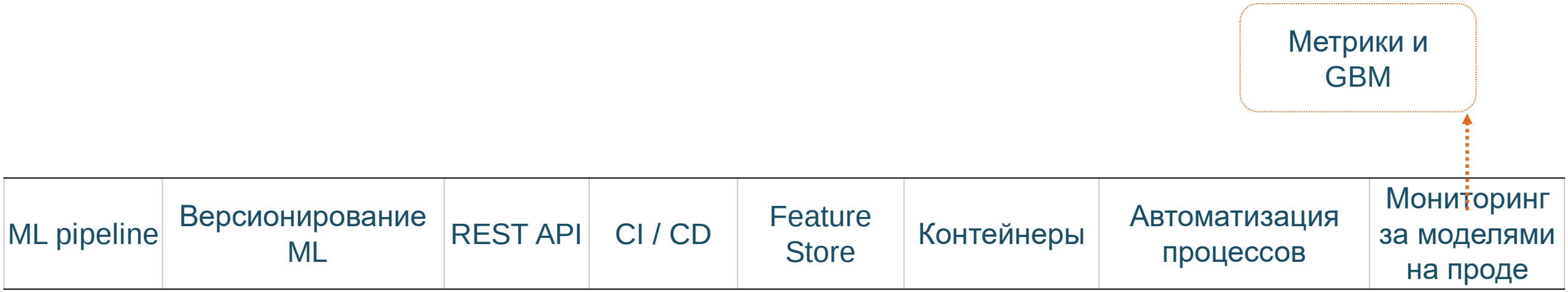
ОБЗОР КУРСА



\+ От простого к сложного, от bonobo к airflow
\+ Пошаговая инструкция применений и вводный курс

! Новый Cron?

ОБЗОР КУРСА



+ Сине-зеленые модели – кто они?
  Что такое «соломенная модель»?
+ Мониторим модель и визуализируем в Streamlit

! Метрики мониторинга модели – не метрики по модели

ML вызовы

Специфичные ошибки

Версионирование

Расширяемость и переиспользуемость

ETL

«Сложность решений»

Работа с конфигурациями

Зависимость от данных

ML вызовы

- Версионирование
- Зависимость от данных
- Расширяемость и переиспользуемость
- ETL
- Работа с конфигурациями
- «Сложность решений»
- Специфичные ошибки

DVC.org

CML

tox

Apache Airflow

pytest

КУДА УХОДЯТ МОДЕЛИ

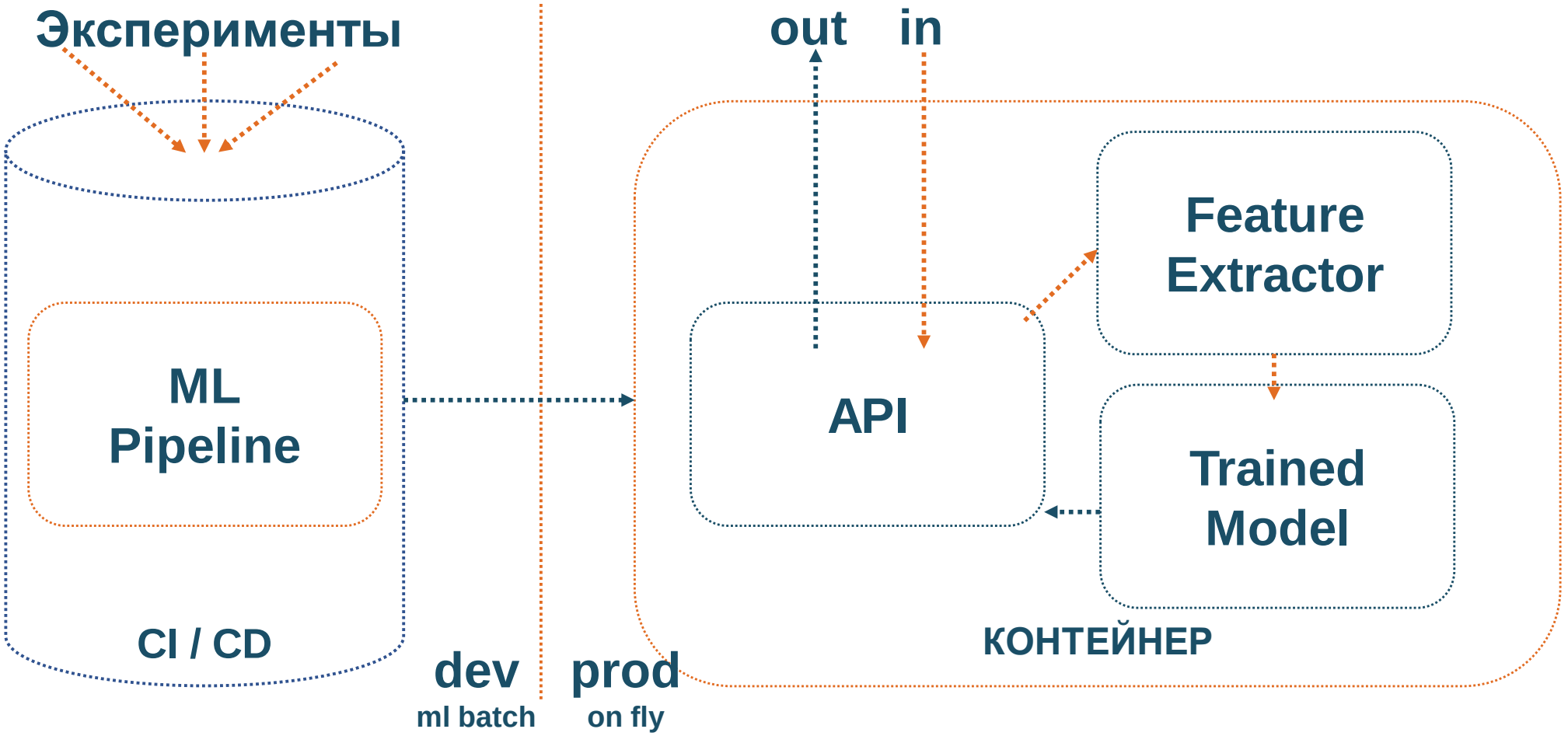
АРХИТЕКТУРА

	Обучение	Предикт	Получить результат предикта	Задержка	Обслуживание системы
DB	batch	batch	DB	большая	просто
API	batch	сразу/по запросу	API	не значительная	не затруднительно
Streaming	streaming	streaming	Queue	низкая	сложно

АРХИТЕКТУРА

	Обучение	Предикт	Получить результат предикта	Задержка	Обслуживание системы
DB	batch	batch	DB	большая	просто
API	batch	сразу/по запросу	API	не значительная	не затруднительно
Streaming	streaming	streaming	Queue	низкая	сложно

АРХИТЕКТУРА



ВИДЫ PIPELINE | ПРОЦЕДУРНЫЙ

```python
# получаем имя
def extract_estimator_name(estimator):

    # проверка естиматор == объект pipeline
    if isinstance(estimator, sklearn.pipeline.Pipeline):
        data_type = type(estimator._final_estimator)

    # проверка естиматор == grid search объект
    elif isinstance(estimator, sklearn.model_selection._search.GridSearchCV):
        # проверка естиматор == объект pipeline
        if isinstance(estimator.best_estimator_, sklearn.pipeline.Pipeline):
            data_type = type(estimator.best_estimator_._final_estimator)

        else:
            data_type = type(estimator.best_estimator_)

    # если это не pipeline и не grid search объект
    else:
        data_type = type(estimator)

    name = ''.join(filter(str.isalnum, str(data_type).split('.')[-1]))

    return name
```

jupyter nbconvert --to script

ВИДЫ PIPELINE | ПРОЦЕДУРНЫЙ

```python
# получаем имя
def extract_estimator_name(estimator):

    # проверка естиматор == объект pipeline
    if isinstance(estimator, sklearn.pipeline.Pipeline):
        data_type = type(estimator._final_estimator)

    # проверка естиматор == grid search объект
    elif isinstance(estimator, sklearn.model_selection._search.GridSearchCV):
        # проверка естиматор == объект pipeline
        if isinstance(estimator.best_estimator_, sklearn.pipeline.Pipeline):
            data_type = type(estimator.best_estimator_._final_estimator)

        else:
            data_type = type(estimator.best_estimator_)

    # если это не pipeline и не grid search объект
    else:
        data_type = type(estimator)

    name = ''.join(filter(str.isalnum, str(data_type).split('.')[-1]))

    return name
```

jupyter nbconvert --to script

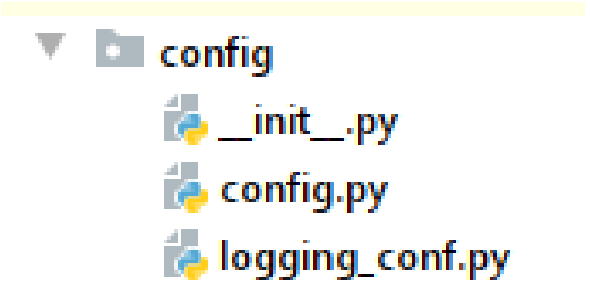
```
jq -j ‘
   .cells |
    map( select(.cell_type == "code")
    | .source + ["\n\n"] )
    | .[][]
    ‘ \
notebook.ipynb > source.py
```

ВИДЫ PIPELINE | SCIKIT-LEARN

```
price_pipe = Pipeline(
    [
        ('categorical_imputer',
            pp.CategoricalImputer(variables=config.CATEGORICAL_VARS_WITH_NA)),

        ('numerical_inputer',
            pp.NumericalImputer(variables=config.NUMERICAL_VARS_WITH_NA)),
        ('scaler', MinMaxScaler()),
        ('Linear_model', Lasso(alpha=0.005, random_state=0))
    ]
)
```

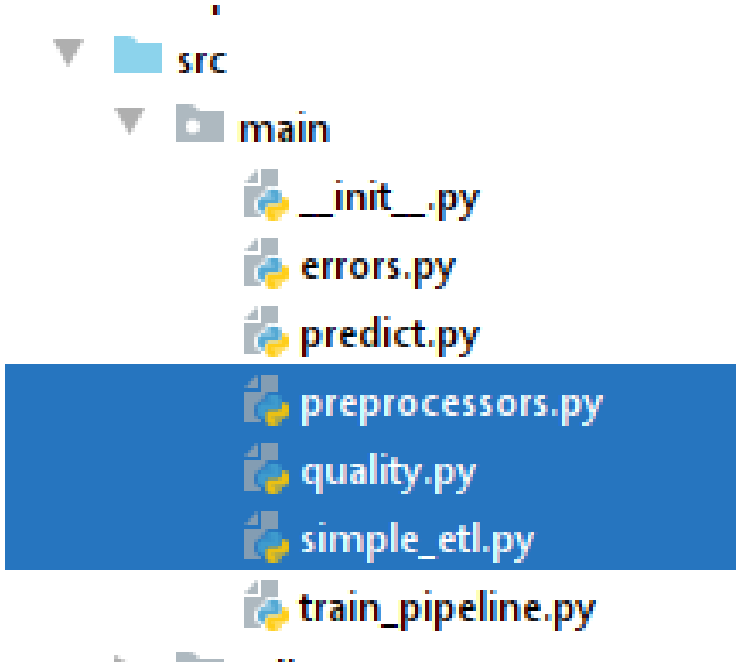
ВИДЫ PIPELINE | CUSTOM

- **Конфигурация**
- **Константы**
- **Пути до файлов**



ВИДЫ PIPELINE | CUSTOM

- **Загрузка данных**
- **Обработка и очистка**
- **Агрегация**
- **Генерация данных**



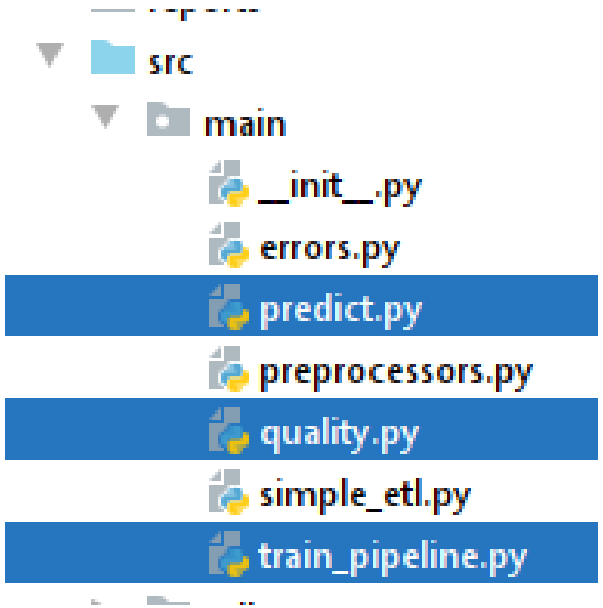
ВИДЫ PIPELINE | CUSTOM

- **Отбор данных**
- **Эксперименты**
- **Отбор модели**

?

ВИДЫ PIPELINE | CUSTOM

- **Обучение**
- **Получение предикта**



ДОБАВИМ ТЕСТОВ

DS TEST

```python
# тесты должны быть осмысленные

def mean(some_list):
  # на примере:
  # тест на пустой список и деление на 0
  # TypeError тест
  # OverFlow тест
  return sum(some_list) / len(some_list)
```

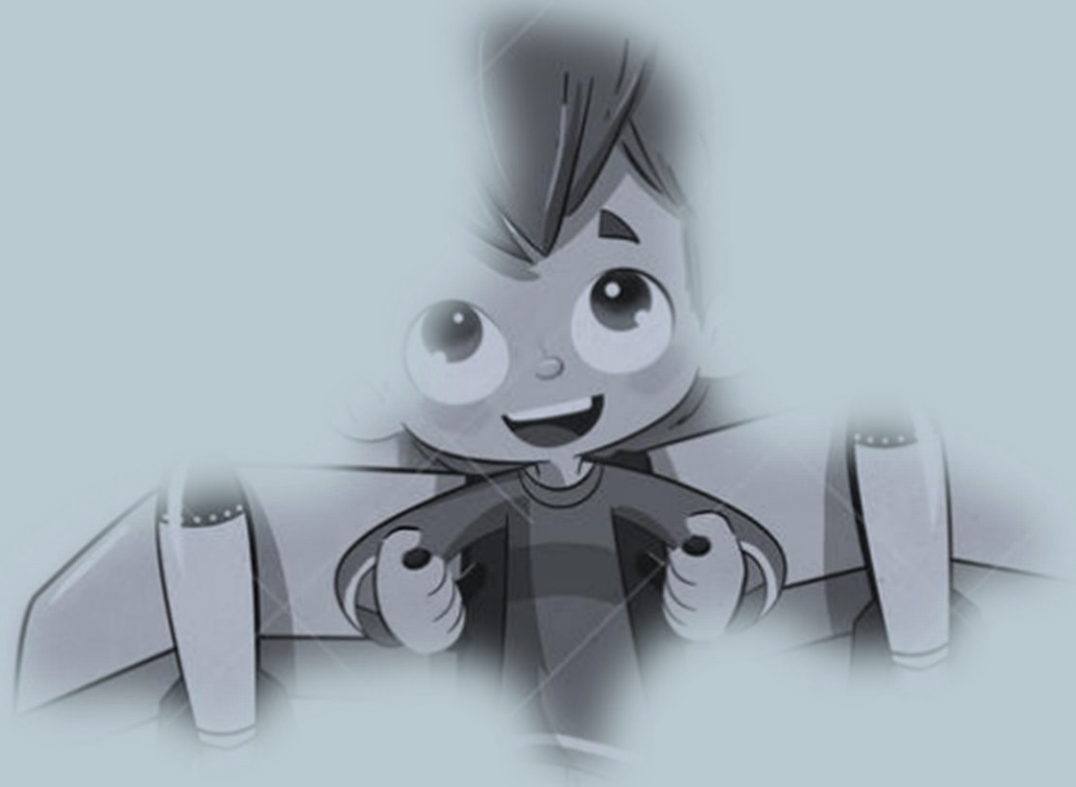
DS TEST

```
# тесты должны быть осмысленные

def mean(some_list):
  # на примере:
  # тест на пустой список и деление на 0
  # TypeError тест
  # OverFlow тест
  return sum(some_list) / len(some_list)

# dev тест
import pytest
def test_dev_mean():
  assert(sum(mean([1,2,3])) / len(3) == 2)
```

DS TEST

```python
# тесты должны быть осмысленные

def mean(some_list):
  # на примере:
  # тест на пустой список и деление на 0
  # TypeError тест
  # OverFlow тест
  return sum(some_list) / len(some_list)

# dev тест
import pytest
def test_dev_mean():
  assert(sum(mean([1,2,3])) / len(3) == 2)

# Но для DS
from hypothesis import given
import hypothesis.strategies as st

@given(st.lists(st.integers()))
def test_ds_mean(some_list):
  assert mean(some_list) == sum(some_list) / len(some_list)
```

ЗАДАНИЕ



ML PIPELINE

- **Реализовать модель в виде procedure pipeline**
- **Реализовать модель в виде scikit-learn pipeline**
- **Реализовать модель в виде custom (oop) pipeline**

- **Сделать 5 – 10 тестов на основные функции и методы реализуемые в вашем pipeline (в одном из вариантов)**