

Pipeline машинного обучения. Особенности архитектурных решений

Пересунько Евгения Олеговна, преподаватель МГТУ им. Н.Э.
Баумана

Структура модуля

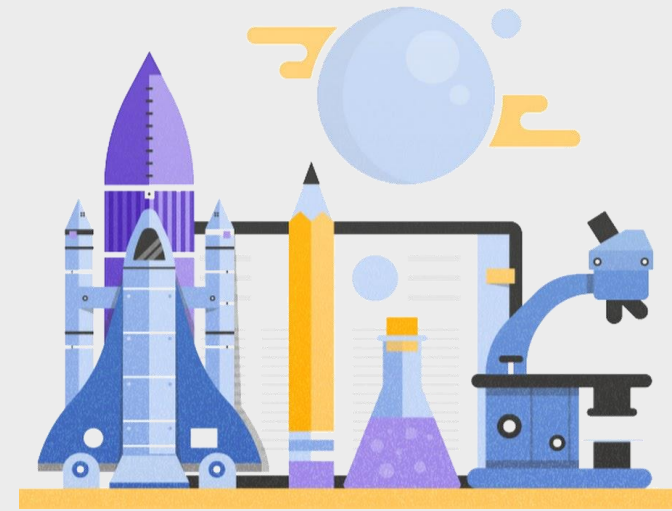
- Pipeline машинного обучения. Особенности архитектурных решений
- Sklearn – основная ML библиотека
- Обучение на размеченных и неразмеченных данных
- Ленивые вычисления

Сегодня на занятии

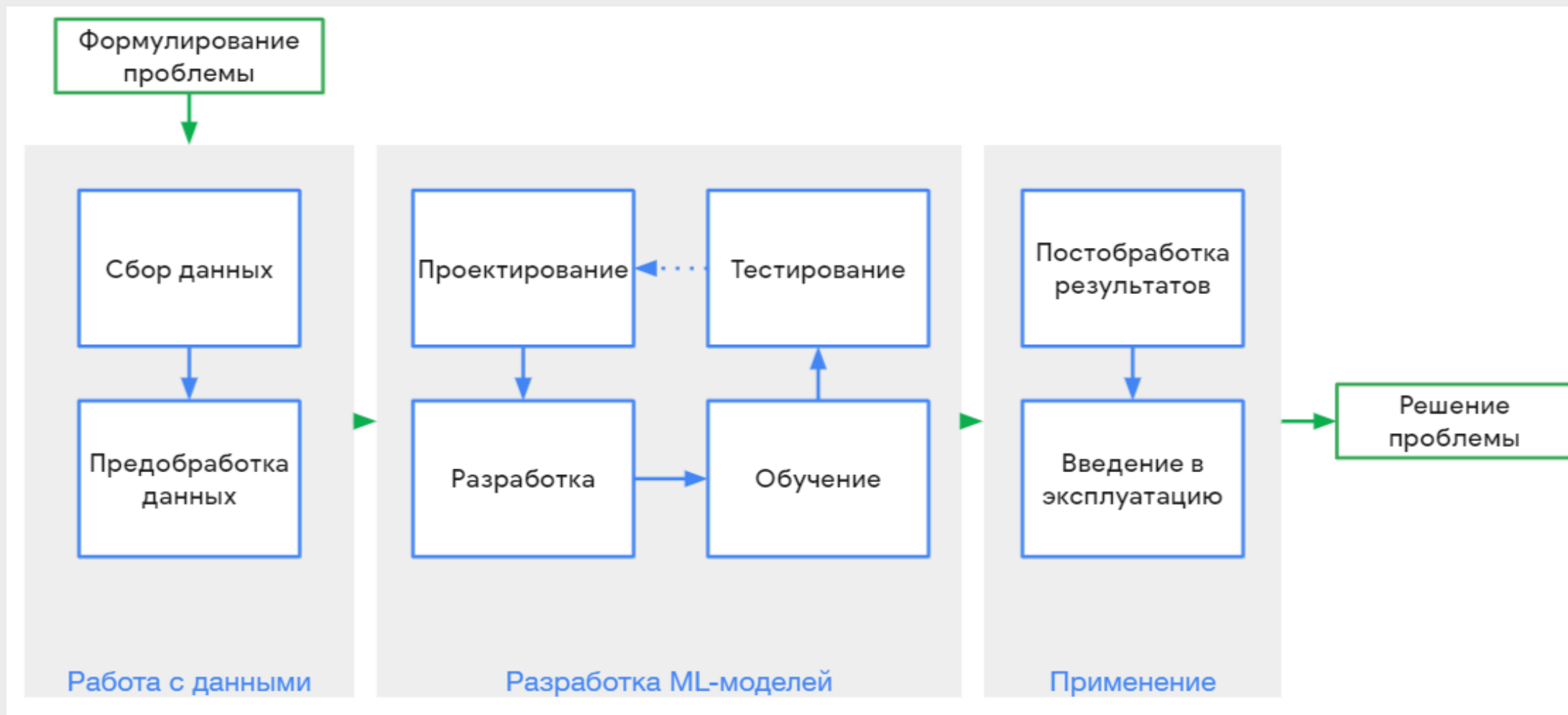
- Основные шаги в проектах по машинному обучению
- Структура проекта по машинному обучению и его архитектура
- Основной pipeline машинного обучения

Как организовать проект по машинному обучению?

Работа над ML-проектом – это объединение двух видов деятельности: разработки программного обеспечения и исследований



Основные шаги при разработке ML-проекта



CRISP-DM

- **CRISP-DM** – Cross-Industry Standard Process for Data Mining
- **CRISP-DM** описывает жизненный цикл исследования данных, состоящий из 6 фаз, от постановки задачи с точки зрения бизнеса до внедрения технического решения.

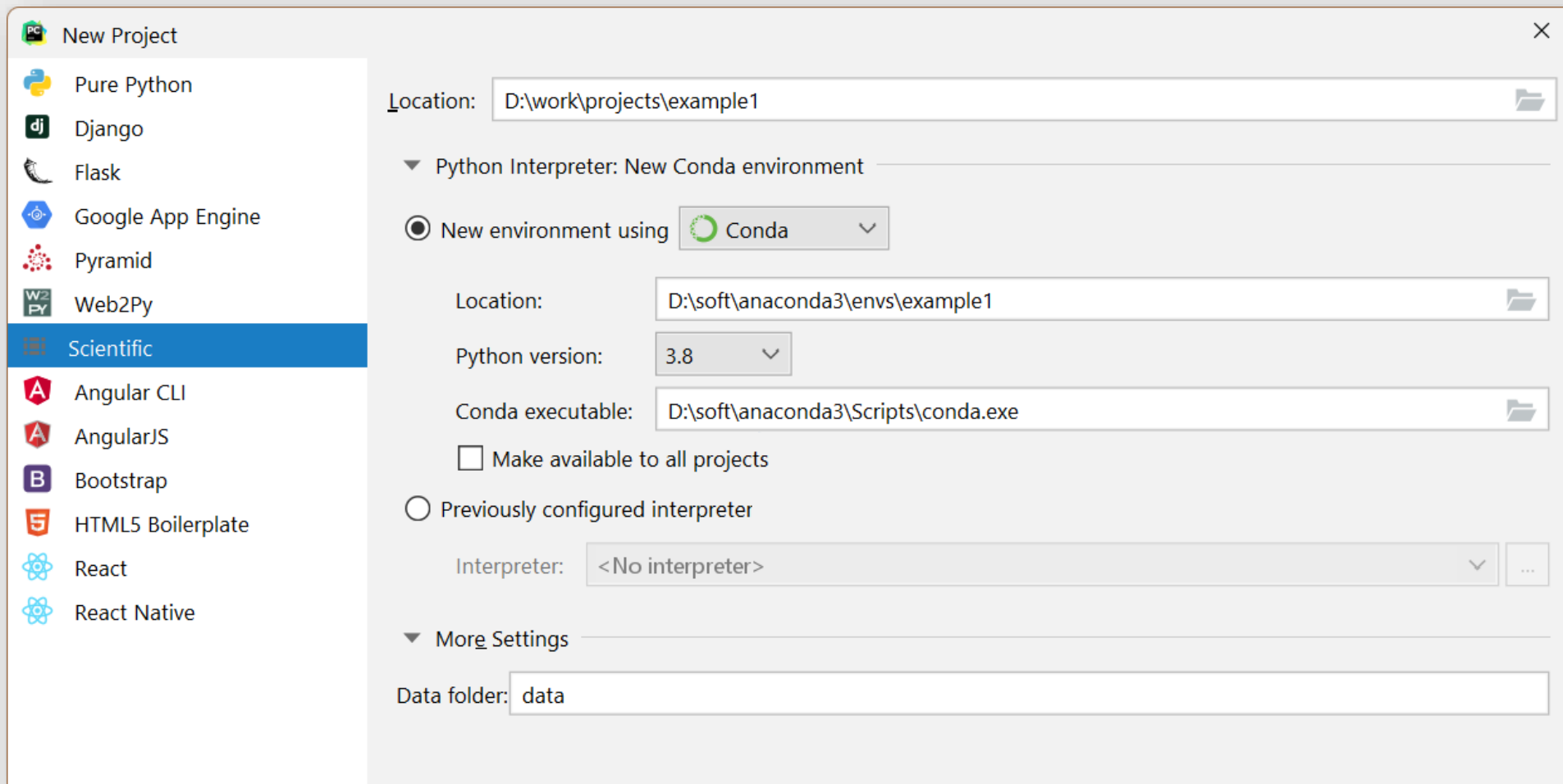
Business Understanding/ Бизнес-анализ	Data Understanding/ Анализ данных	Data Preparation/ Подготовка данных	Modeling/ Моделирование	Evaluation/ Оценка решения	Deployment/ Внедрение
Determine Business Objectives/ Определение бизнес-целей	Collect Initial Data/ Сбор данных	Select Data/ Выборка данных	Select Modeling Techniques/ Выбор алгоритмов	Evaluate Results/ Оценка результатов	Plan Deployment/ Внедрение
Assess Situation/ Оценка текущей ситуации	Describe Data/ Описание данных	Clean Data/ Очистка данных	Generate Test Design/ Подготовка плана тестирования	Review Process/ Оценка процесса	Plan Monitoring and Maintenance/ Планирование мониторинга и поддержки
Determine Data Mining Goals/ Определение целей аналитики	Explore Data/ Изучение данных	Construct Data/ Генерация данных	Build Model/ Обучение моделей	Determine Next Steps/ Определение следующих шагов	Produce Final Report/ Подготовка отчета
Produce Project Plan/ Подготовка плана проекта	Verify Data Quality/ Проверка качества данных	Integrate Data/ Интеграция данных	Assess Model/ Оценка качества моделей		Review Project/ Ревью проекта
		Format Data/ Форматирование данных			



Из чего состоит проект по машинному обучению?

- Основные элементы проекта на Python:
- Исходный код:
 - Скрипты на Python - основные модули программы
 - «Тетрадки» Jupyter Notebook - эксперименты, исследование данных и т. п.
- Наборы данных
- Модели
- Окружение

Создание исследовательского проекта в PyCharm



The image shows the 'New Project' dialog in PyCharm. On the left is a sidebar with project templates: Pure Python, Django, Flask, Google App Engine, Pyramid, Web2Py, Scientific (highlighted), Angular CLI, AngularJS, Bootstrap, HTML5 Boilerplate, React, and React Native. The main area is for configuring the 'Scientific' project. It includes a 'Location' field set to 'D:\work\projects\example1'. Under 'Python Interpreter: New Conda environment', the 'New environment using Conda' option is selected. This section has fields for 'Location' (D:\soft\anaconda3\envs\example1), 'Python version' (3.8), and 'Conda executable' (D:\soft\anaconda3\Scripts\conda.exe). There is an unchecked checkbox for 'Make available to all projects' and an unselected radio button for 'Previously configured interpreter'. The 'Interpreter' dropdown is set to '<No interpreter>'. A 'More Settings' section is collapsed. At the bottom, the 'Data folder' is set to 'data'.

New Project

Pure Python

Django

Flask

Google App Engine

Pyramid

Web2Py

Scientific

Angular CLI

AngularJS

Bootstrap

HTML5 Boilerplate

React

React Native

Location: D:\work\projects\example1

Python Interpreter: New Conda environment

☒ New environment using Conda

Location: D:\soft\anaconda3\envs\example1

Python version: 3.8

Conda executable: D:\soft\anaconda3\Scripts\conda.exe

☐ Make available to all projects

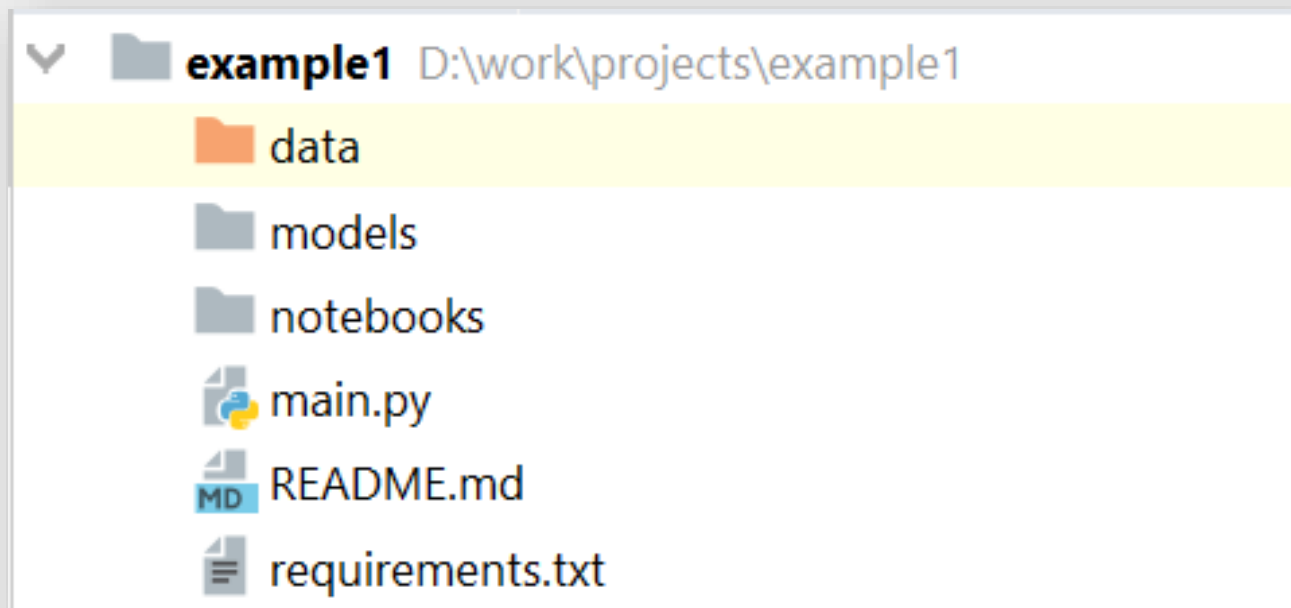
☐ Previously configured interpreter

Interpreter: <No interpreter>

More Settings

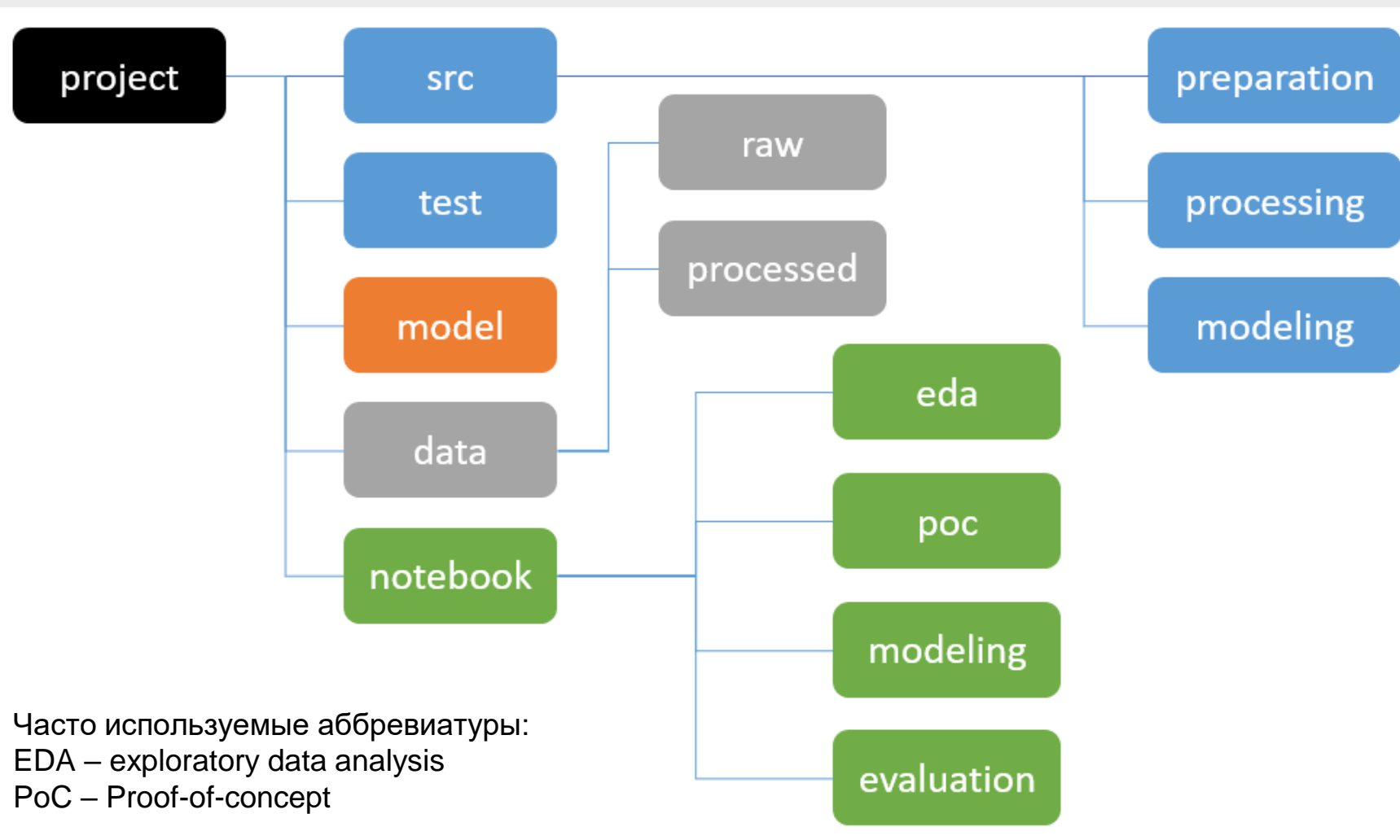
Data folder: data

Структура базового проекта



- README.md – файл с инструкцией для запуска и любой важной информацией
- requirements.txt – файл, в котором перечислены все необходимые зависимости
- файлы *.py – модули программы
- data – папка с наборами данных
- models – папка с моделями и их параметрами
- notebooks – папка с «тетрадками»

Расширенная структура проекта



Часто используемые аббревиатуры:
EDA – exploratory data analysis
PoC – Proof-of-concept

Jupyter Notebook

notebook - папка с файлами Jupyter Notebook, которая может содержать в себе следующие блокноты или подпапки:

- **eda [Exploratory Data Analysis (Data Exploration)]** - исследование данных.
- **poc [Proof-of-Concept]** - проверка концепции. Разработка некоторого прототипа или его частей, которые отражают основную идею. Проверка осуществимости и применимости различных методов
- **modeling** - построение модели и ее обучение
- **evaluation** - оценка качества модели

Модели и данные

- **model** - папка с моделями и мета-данными. Внутри может быть создана папка **logs** с логами обучения
- **data** - папка с данными, включающая в себя папки:
 - **raw** - сырые данные, то есть данные в том виде, в котором были получены
 - **processed** - данные после различных преобразований (удаление выбросов, заполнение пропусков и т.д.)
- **[примечание]** можно разбить эти папки на train и test

Исходный код программы

- **src (source code)** – папка с исходным кодом программы, которая может делиться на следующие подпапки:
 - **preparation** - подготовка данных, например, выгрузка из базы данных, чтение из файла и т.д. Все то, что нужно сделать перед обработкой
 - **processing** - обработка данных: чистка, трансформации, преобразования
 - **modeling** - построение модели
- **[примечание]** В данной структуре удобно создавать файлы **utils.py** (для всяких вспомогательных функций), **train.py** (для обучения модели), **predict.py** (для получения предсказаний)
- **test** - папка с тестами (для кода программы, а не модели!)

Завершение первой части лекции

В первой части мы познакомились со структурой проекта, во второй - пайплайн

Pipeline или Workflow?

- **Пайплайн (англ. pipeline)** — это процесс разработки (подготовки, производства), программный конвейер.
- **Поток работ (англ. workflow)** — это графическое представление потока задач в процессе и связанных с ним подпроцессов, включая специфические работы, информационные зависимости и последовательность решений и работ.

Отличие пайплайна от потока работ

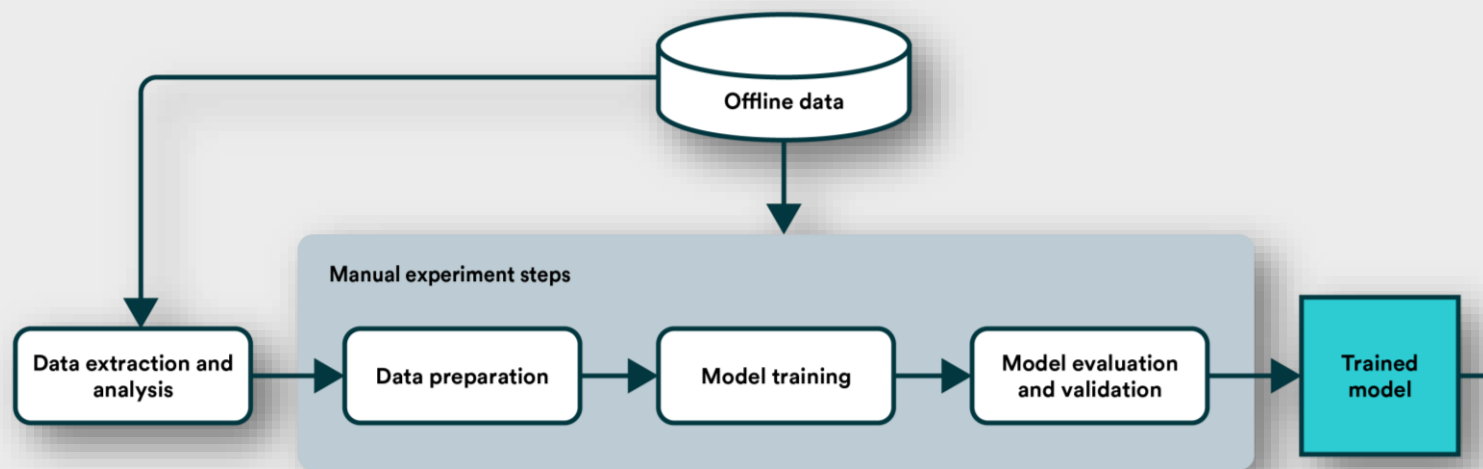


Рис. 1 - Pipeline

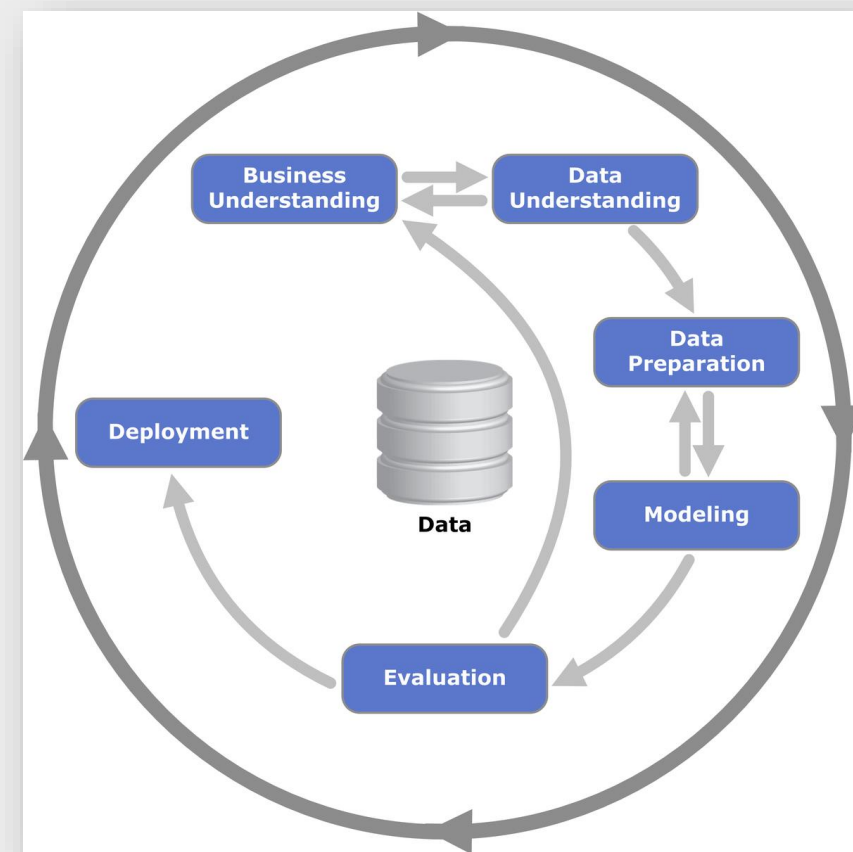
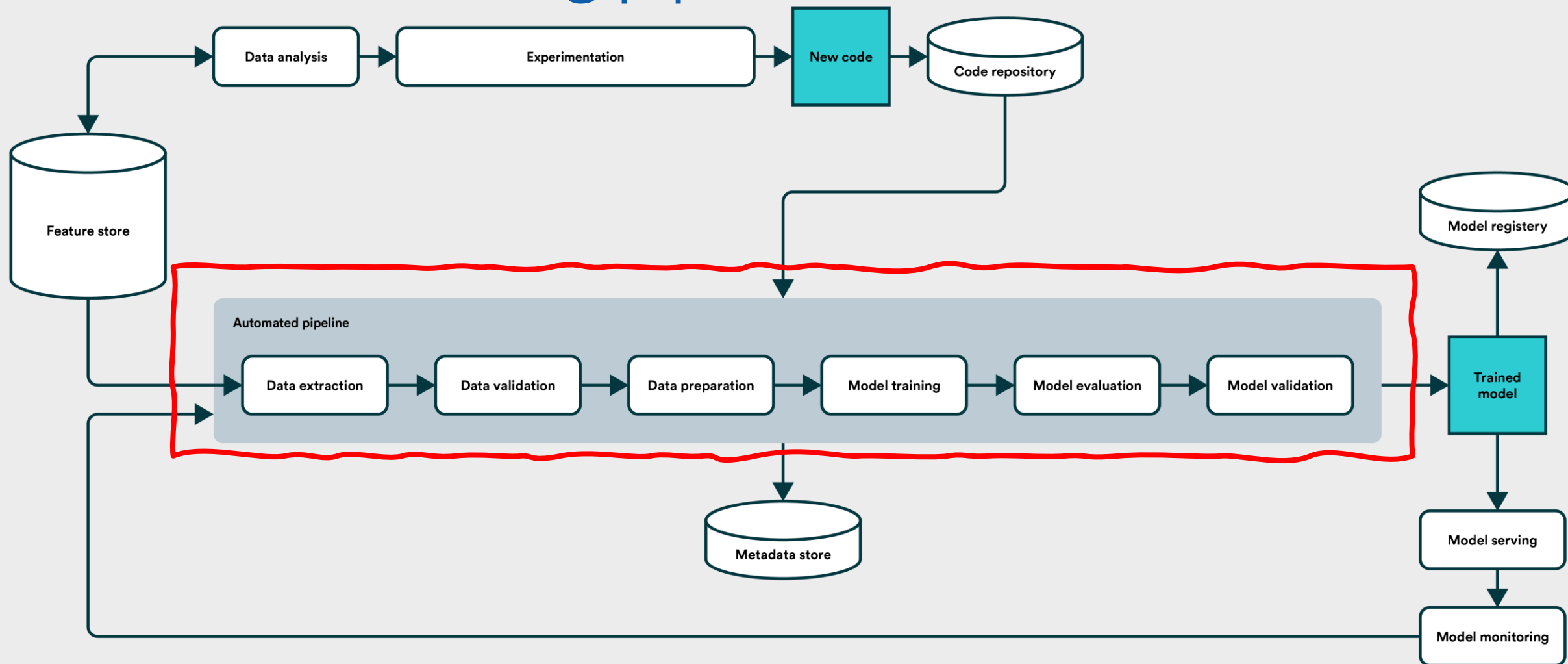


Рис. 2 - Workflow

Какие бывают пайплайны?

- Пайплайн – это последовательность действий, где каждый следующий шаг использует результат работы предыдущего.
- В проекте по машинному обучению можно выделить следующие пайплайны:
 - DevOps (MLOps) pipeline – это конвейер для организации запуска разработанного приложения «в продакшн»
 - ML pipeline – это конвейер преобразования входных данных в выходные

Machine Learning pipeline



Инструменты для ML-пайплайнов

Популярные

Искусственный интеллект и машинное обучение

Среда выполнения приложений

Контейнеры

Гибридные и многооблачные решения

Интернет вещей (IoT)

[Все продукты \(более 200\) >](#)



Создание, обучение и развертывание моделей машинного обучения

Машинное обучение
Azure



Добавление в приложения возможностей с помощью API

Azure Cognitive Services



Быстро развертывайте решения на базе ИИ для распространенных бизнес-процессов

Сервисы прикладного ИИ Azure



Добавление в приложения поиска содержимого на основе ИИ

Когнитивный поиск
Azure



Создание ботов и их подключение к различным каналам

Службы Azure Bot



Применение расширенных языковых моделей при различных вариантах использования

Служба Azure OpenAI

Microsoft Azure - облачная платформа, предоставляющая возможность разработки, выполнения приложений и хранения данных



ОБРАЗОВАТЕЛЬНЫЙ
ЦЕНТР МГТУ им. Н. Э. Баумана

Возможности Microsoft Azure

Поддержка сквозного жизненного цикла машинного обучения (ML)

Подготовка данных

Сборка и обучение моделей

Проверка и развертывание

Управление и мониторинг



Маркировка данных

Label training data and manage labeling projects.

Подготовка данных

Интеграция с модулями аналитики для исследования и подготовки данных.

Наборы данных

Доступ к данным, создание наборов данных и предоставление к ним общего доступа.

Пайплайны в Microsoft Azure

Подготовка данных

Сборка и обучение моделей

Проверка и развертывание

Управление и мониторинг



Управляемые конечные точки

Используйте развертывание одним щелчком для пакетного вывода и вывода в режиме реального времени.

Гибридная и многооблачная среда

Обучайте и развертывайте модели в локальной и многооблачной среде.

Pipelines and CI/CD

Автоматизируйте рабочие процессы машинного обучения.

Optimize models

Ускорьте обучение и вывод, сократив затраты, с помощью среды выполнения ONNX.

Предварительно созданные образы

Получайте доступ к образам контейнеров с платформами и библиотеками для вывода.

Model repository

Share and track models and data.



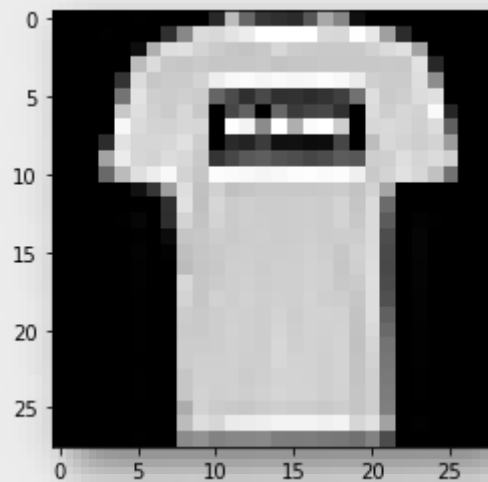
**ОБРАЗОВАТЕЛЬНЫЙ
ЦЕНТР** МГТУ им. Н. Э. Баумана

Другие инструменты организации ML-пайплайнов

- **TensorFlow Extended (TFX)** - это комплексная платформа для развертывания ML-конвейеров (<https://www.tensorflow.org/tfx?hl=ru>)
- **Valohai** – MLOps платформа (<https://valohai.com/>)
- **Apache Spark (Spark ML, MLlib)** - фреймворк с открытым исходным кодом для реализации распределённой обработки данных (<https://spark.apache.org/mlib/>)
- **Apache Airflow** – инструмент для управления конвейерами на базе Python и SQL (<https://airflow.apache.org/>). Пример «надстройки» - astronomer.io (<https://www.astronomer.io/>)

Создание пайплайна с помощью библиотеки scikit-learn

- Пример задачи: определить по изображению предмета одежды его наименование



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

- Вариант решения: обрезать черные границы по краям и обучить классификатор

Создание пайплайна с помощью библиотеки scikit-learn

- Sklearn-pipeline позволяет автоматизировать процесс работы с данными

```
In [105]: pipeline = Pipeline([
            ('preprocessing', FunctionTransformer(crop_borders)),
            ('clf', KNeighborsClassifier(n_neighbors=5))
        ])
            pipeline.steps

Out[105]: [('preprocessing',
            FunctionTransformer(func=<function crop_borders at 0x0000026EA329FF70>)),
            ('clf', KNeighborsClassifier())]
```


Классификация текстов с помощью пайплайна sklearn

```
: from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC

pipe = Pipeline([
    ('vectorize', CountVectorizer()),
    ('tfidf', TfidfVectorizer()),
    ('classify', LinearSVC())
])

X_train, X_test, y_train, y_test = train_test_split(X, y)
pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)
print(classification_report(y_test, y_pred))
```

Завершающая отбивка

Итак, сегодня мы с вами изучили тему Pipeline машинного обучения. Особенности архитектурных решений

Я познакомила вас с

Теперь вы знаете

Умеете

Это только начало. Изучайте курс дальше. До встречи на следующих уроках



edu.bmstu.ru

+7 (495) 120-99-76

edu@bmstu.ru

Москва, 2-ая Бауманская, д. 5, с. 1