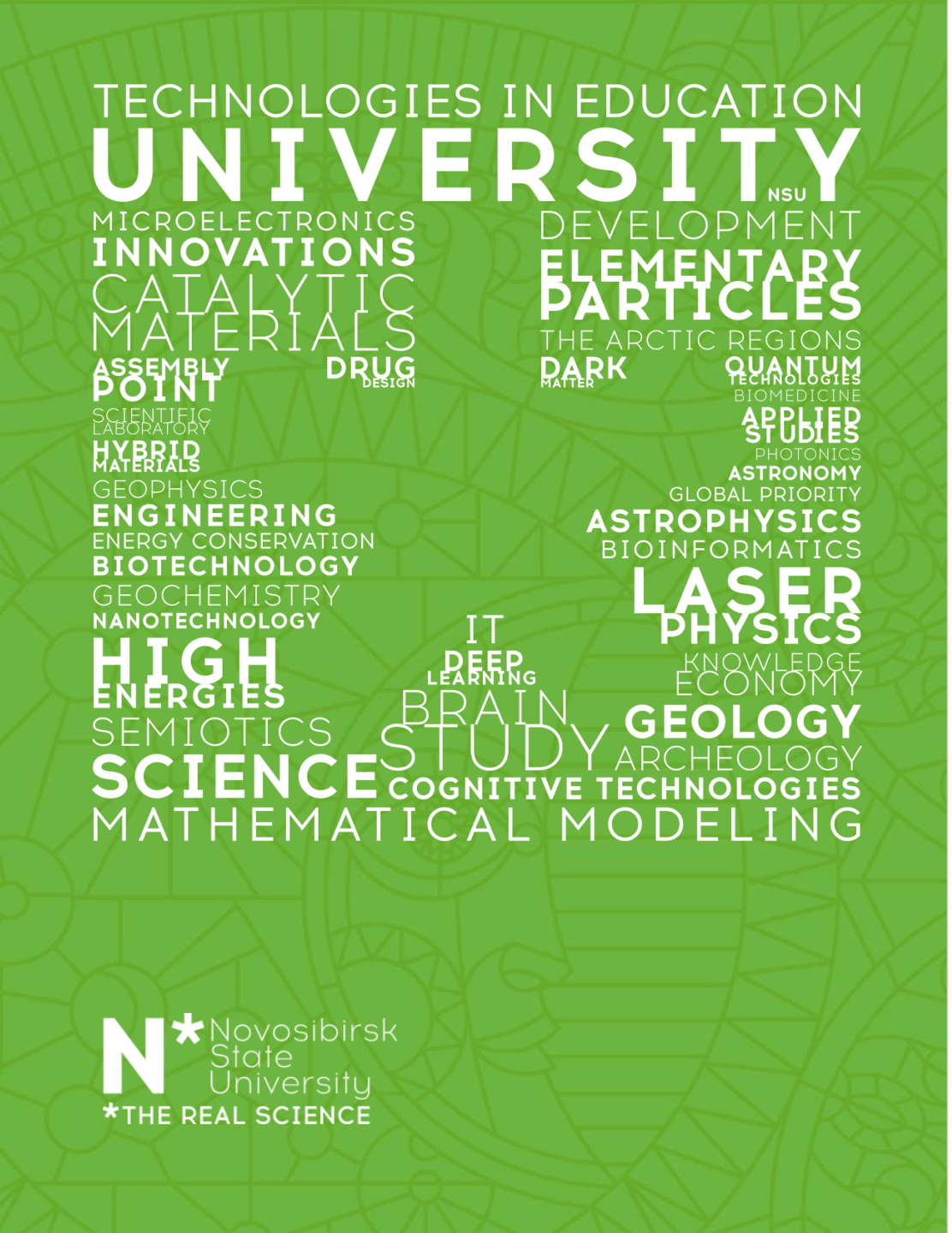


Машинное обучение
Семинар 2

Глушенко Андрей Валерьевич
Институт Интеллектуальной Робототехники

Вызовы в ИИ/ML-проектах

- ✓Создание и управление конвейерами — сложная задача
- ✓Непрерывный итеративный процесс, оптимизация по метрикам
- ✓Со временем данные меняются, происходит дрейф модели.
- ✓Отслеживание экспериментов затруднено
- ✓Артефакты модели теряются
- ✓Разнообразные среды развертывания
- ✓Что происходит при масштабировании проекта?

DevOps | MLOps

CAMS

CI/CD

IaC

Triggering

DevOps

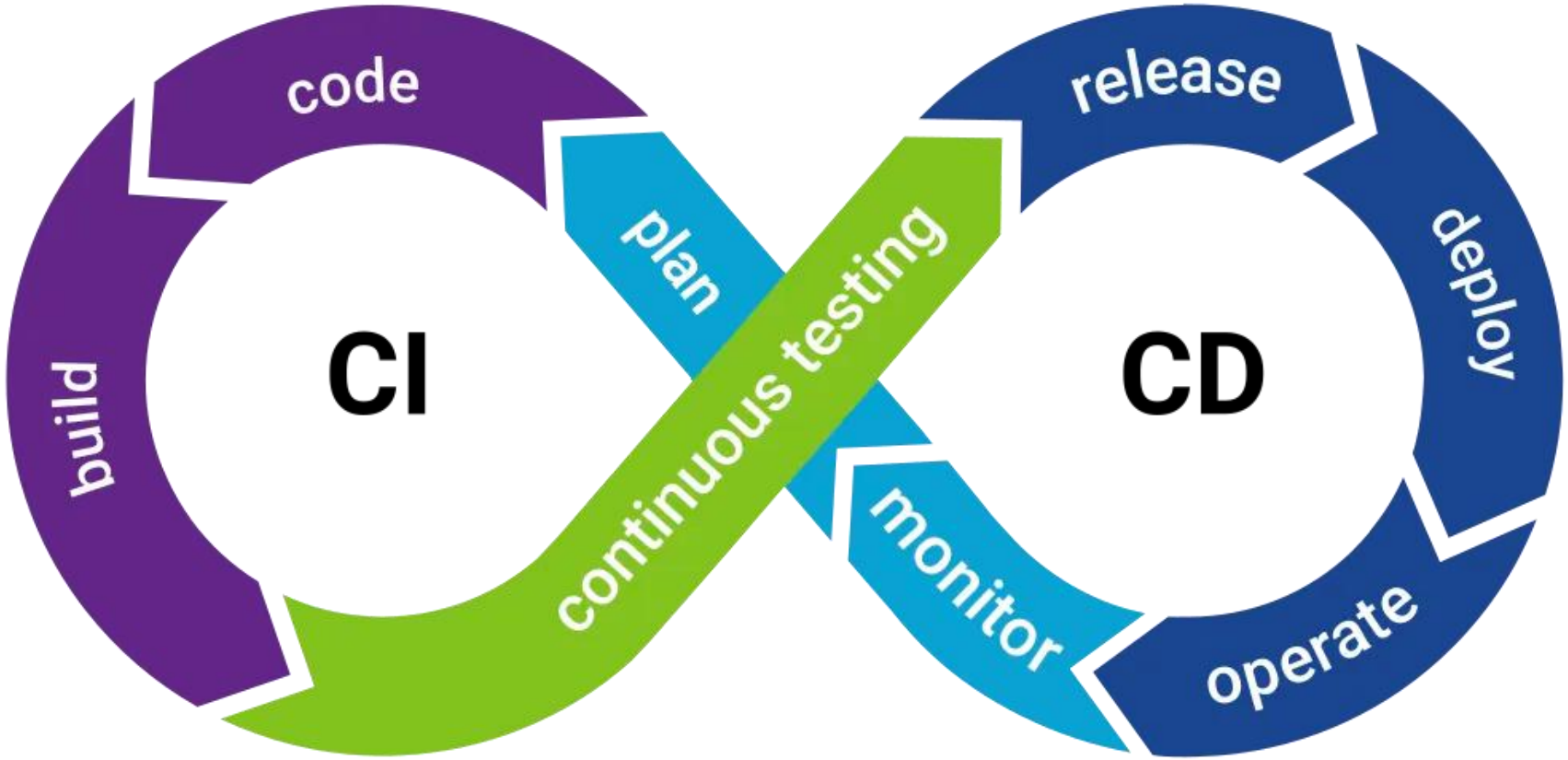
- Раньше разработка была похожа на эстафету: программисты писали код и «перебрасывали его через забор » админам
- Если на сервере всё ломалось, админы винили кривой код, а разработчики — кривую настройку серверов
- **DevOps объединяет их в одну команду**, которая вместе отвечает за продукт от первой строчки кода до работы у пользователя

Три кита DevOps: CAMS

- **Culture:** Общая ответственность. Сломалось? Чиним вместе.
- **Automation:** Всё, что можно сделать дважды, должен делать скрипт.
- **Measurement:** Сбор метрик — мы должны знать всё о здоровье системы.
- **Sharing:** Знания и инструменты общие для всей команды.

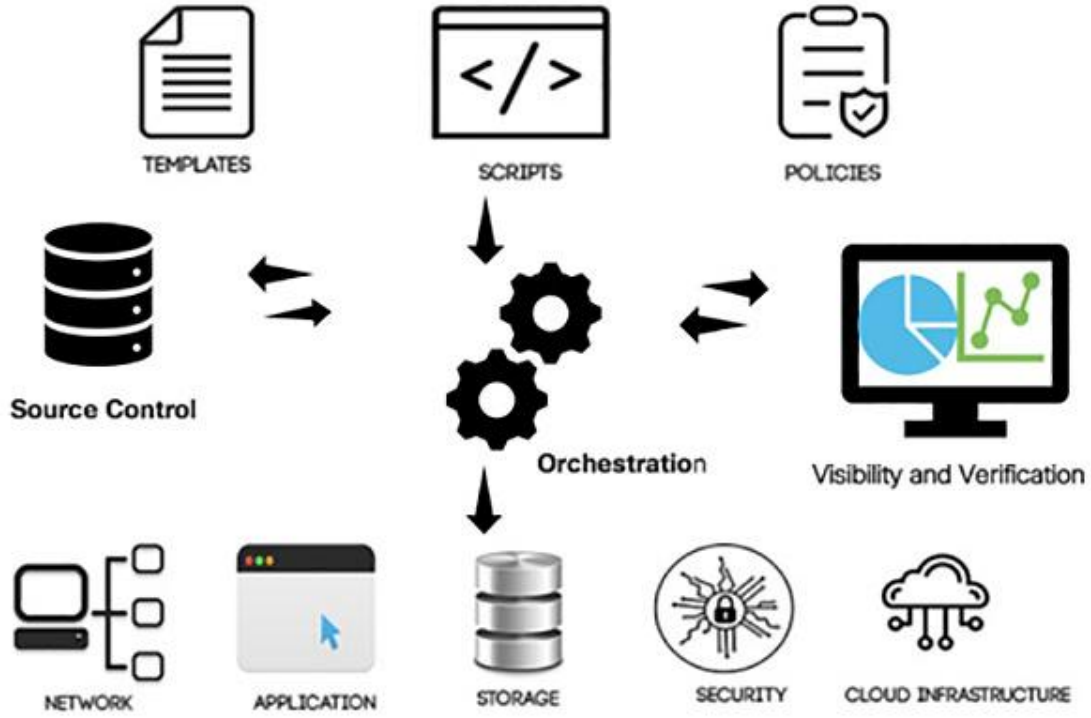
Три кита DevOps: CI/CD

- **CI (Continuous Integration):** Как только программист сохраняет код, он автоматически проверяется тестами.
- **CD (Continuous Delivery):** Если тесты прошли, код автоматически готовится к выкатке (деплою).



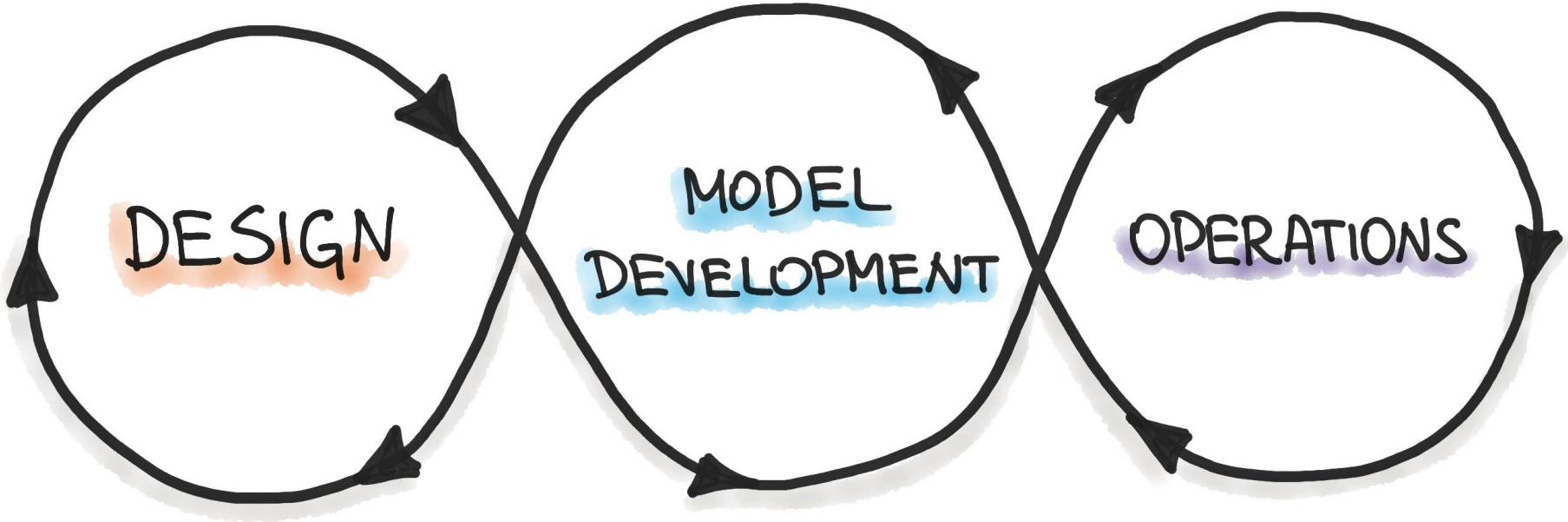
Три кита DevOps: IaC (Инфраструктура как код)

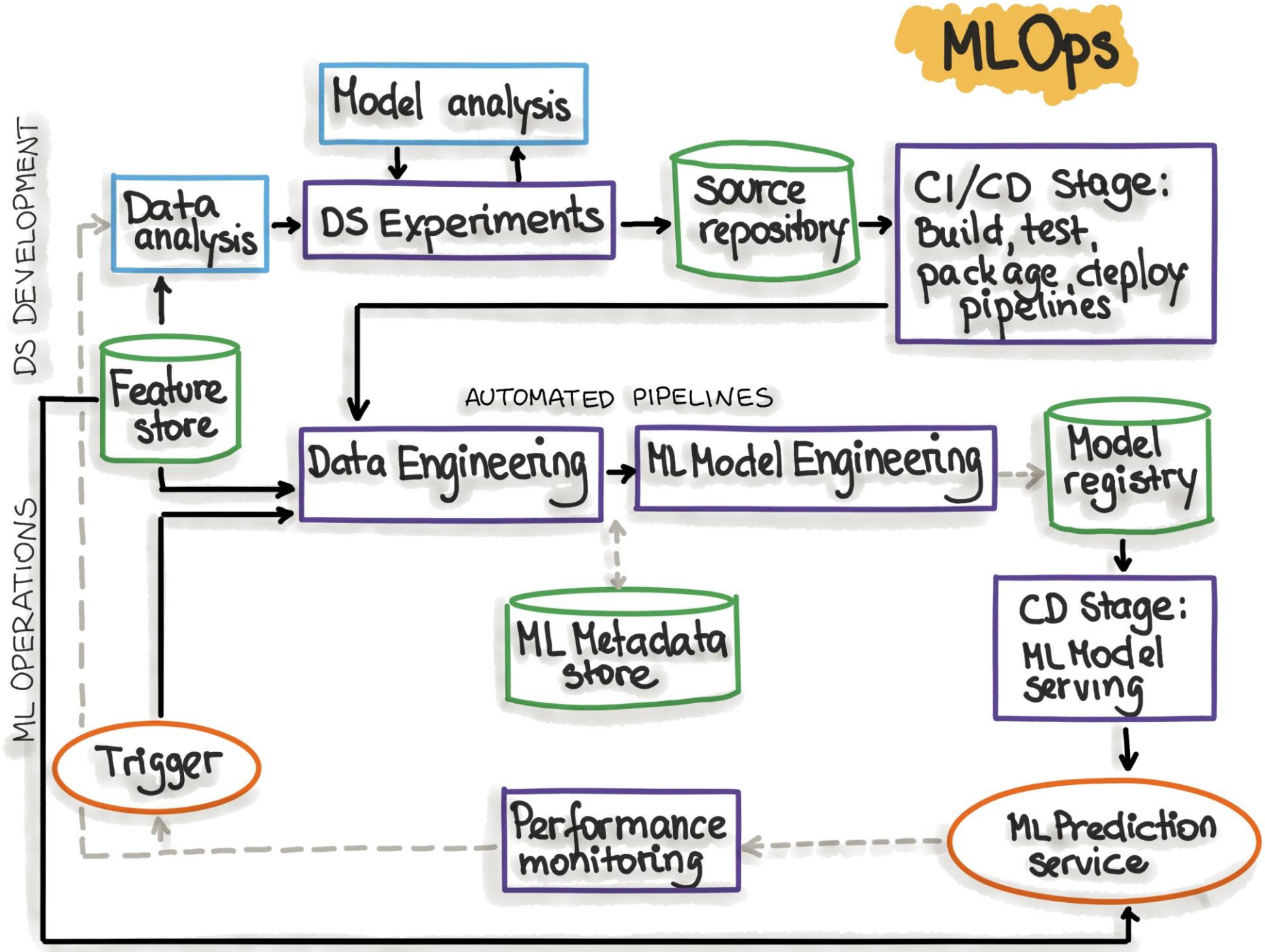
- Вместо того чтобы вручную настраивать сервер кнопками, инженер пишет текстовый файл-конфигурацию. Это позволяет «поднять» 100 одинаковых серверов за пару минут одной командой.



Что такое MLOps?

- это набор практик по автоматизации жизненного цикла моделей машинного обучения: от подготовки данных до деплоя и мониторинга. Если классический DevOps управляет кодом, то MLOps управляет **кодом, данными и самой моделью**.




MLOps
Model analysis
Data analysis
DS Experiments
Source repository
CI/CD Stage: Build, test, package, deploy pipelines
DS DEVELOPMENT
Feature store
AUTOMATED PIPELINES
Data Engineering
ML Model Engineering
Model registry
ML Metadata store
CD Stage: ML Model serving
ML OPERATIONS
Trigger
Performance monitoring
ML Prediction service

Triggering в MLOps

— это автоматический запуск пайплайна (цепочки действий) в ответ на определенное событие. Основные виды триггеров:

- **По расписанию (Schedule-based):** Самый простой вариант. Модель переобучается раз в неделю или месяц, чтобы всегда оставаться актуальной.
- **По новым данным (Data-driven):** Пайплайн запускается, как только в хранилище (например, S3 или Feature Store) накопился определенный объем свежих данных.
- **По деградации метрик (Performance-based):** Система мониторинга видит, что точность модели упала ниже критического порога (например, Precision стал < 80%), и автоматически дает сигнал на переобучение.
- **По дрейфу данных (Drift-based):** Триггер срабатывает, если статистические свойства входящих данных сильно изменились по сравнению с теми, на которых модель обучалась изначально.
- **По событию в коде (Event-driven):** Классический запуск при пуше нового кода или изменении гиперпараметров в Git.

Важность для нашего курса

- Вам необходимо совместно с одногруппниками разработать общее решение с использованием ML-моделей для одного из кейсов
- Решение должно представлять собой MVP
- Воспроизводимость решения задачи (кейса)
- Подготовка материалов по вашему проекту (кейсу) для презентации компании и кафедре

Bitbucket

GitLab

GitHub

git

Git - система контроля версий
(VCS = Version Control System)

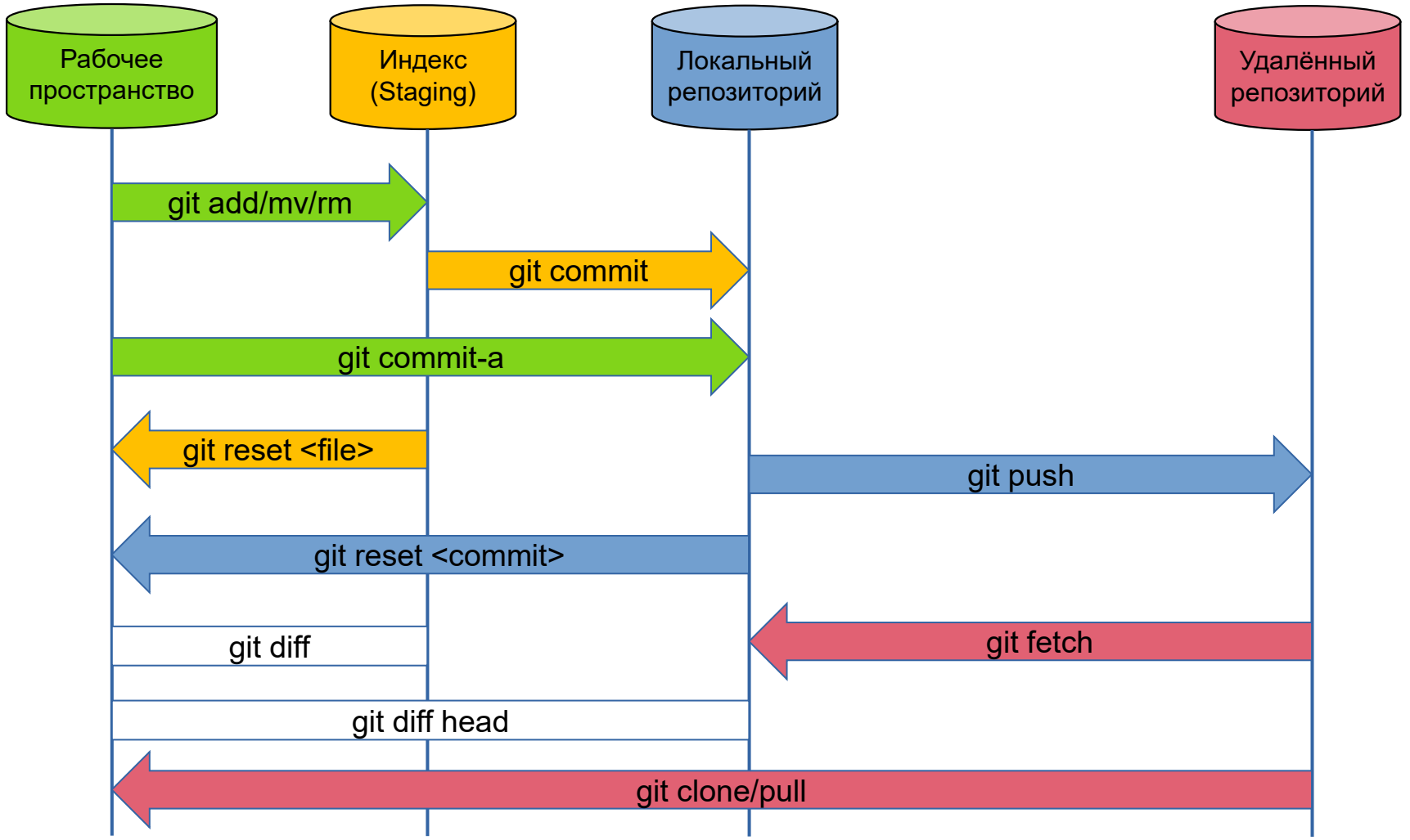
Репозиторий

Репозиторий (репо) — это, по сути, продвинутая папка с проектом, которая находится под управлением системы контроля версий (Git).

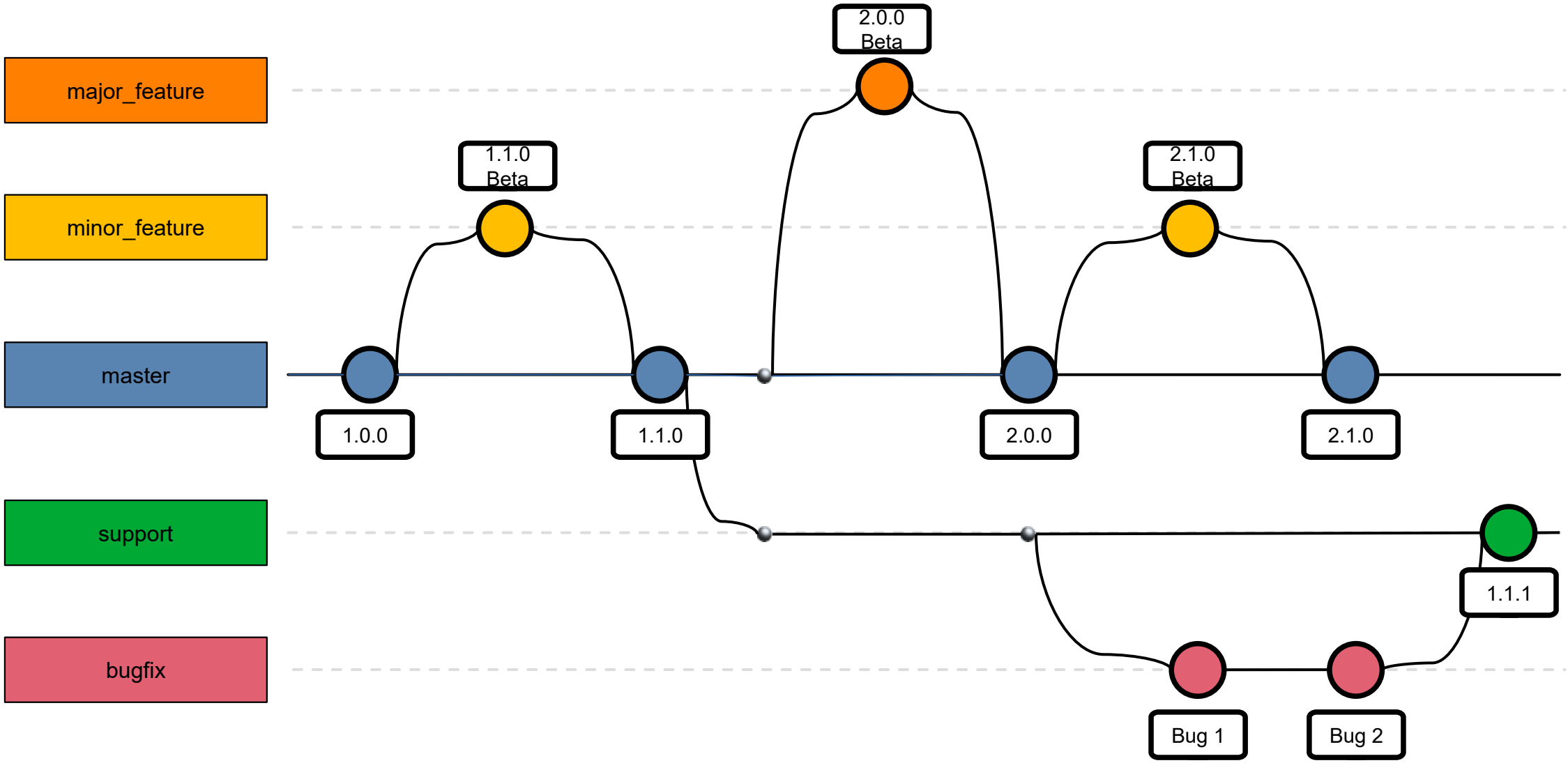
Из чего состоит репозиторий:

- **Рабочий каталог:** Сами файлы, редактируемые прямо сейчас.
- **Служебная папка .git:** Скрытая директория, где лежит вся «магия» — история коммитов, настройки и логи. Если её удалить, репозиторий превратится в обычную папку.
- **История коммитов:** Цепочка снимков проекта. Всегда можно «отмотать» состояние папки к любому коммиту из прошлого.
- **Ветки (Branches):** Возможность держать в одной папке несколько версий проекта (например, одна ветка — стабильный сайт, вторая — эксперимент с новым дизайном).

Архитектура



Ветки


major_feature
minor_feature
master
support
bugfix
2.0.0
Beta
1.1.0
Beta
2.1.0
Beta
1.0.0
1.1.0
2.0.0
2.1.0
1.1.1
Bug 1
Bug 2

Базовые команды git

Новый репозиторий git

- git init – инициализируем новый репо
- git remote add <NAME> <URL> - добавить удалённый репозиторий

Протоколы передачи для git

- FILE – прямой доступ к файлам репозитория
- SSH – у нас есть доступ к файлам на сервере через ssh
- HTTP(S) – используем http для получения/передачи

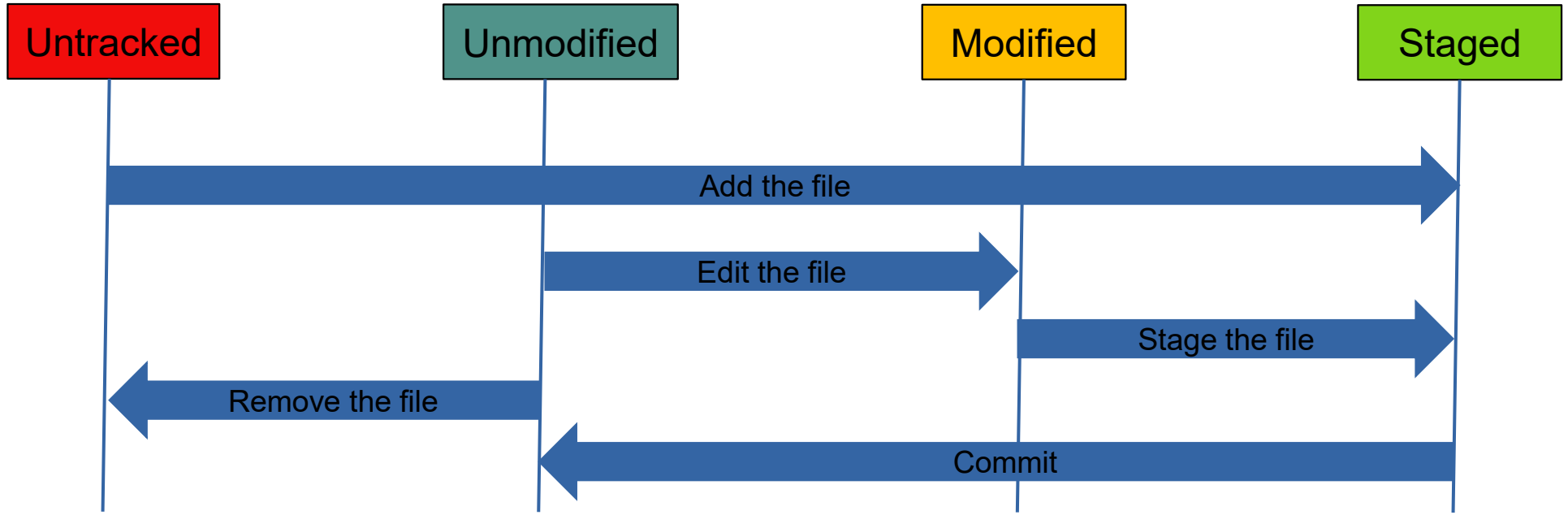
Существующий репозиторий

- git clone <URL> - клонируем удалённый репозиторий git
- git clone <URL> <PATH> - клонируем репо в указанную директорию

Базовые команды git

Добавление изменений в git

- git add <FILENAME> – добавить файл в индекс
- git commit -m “<MESSAGE>” – закоммитить изменения
- git status – текущий статус файлов в репозитории



Работа с файлами

Удаление файлов

- git rm <FILENAME> – удалить файл
- git rm --cached <FILENAME> - удалить файл только из индекса

Перемещение файлов

- git mv <FILE_FROM> <FILE_TO> - переместить/переименовать файл

Работа с ветками

Создание ветки

- git branch <NAME> – создать ветку
- git branch -a – получить список локальных веток

Переключение веток

- git status – узнать текущую ветку
- git checkout <NAME> - переключиться на ветку
- git checkout -b <NAME> - создать новую ветку и переключиться на неё
- git switch <NAME> - переключиться на ветку
- git switch -c <NAME> - создать ветку и переключиться на неё

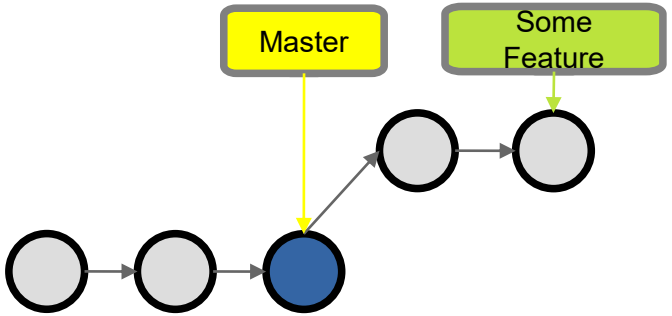
Слияние веток

- git merge <BRANCH> – слияние ветки BRANCH в текущую ветку
- git merge –ff <BRANCH> – “fast-forward” текущая ветка в ветку BRANCH, если BRANCH продолжение текущей ветки
- git merge –no-ff <BRANCH> - создать коммит, чтобы заммерджить ветку BRANCH в текущую
- git merge --ff-only <BRANCH> - “fast-forward” ветки в BRANCH, если возможно

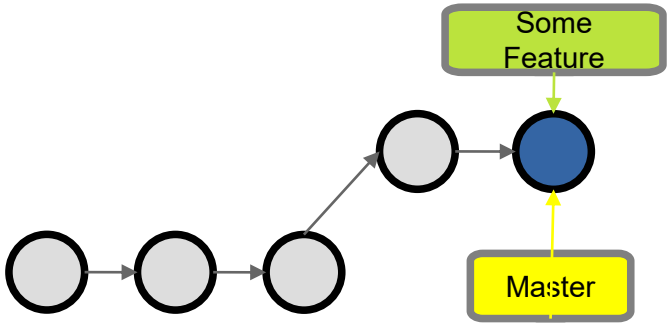
Слияние веток

fast-forward merge

До слияния

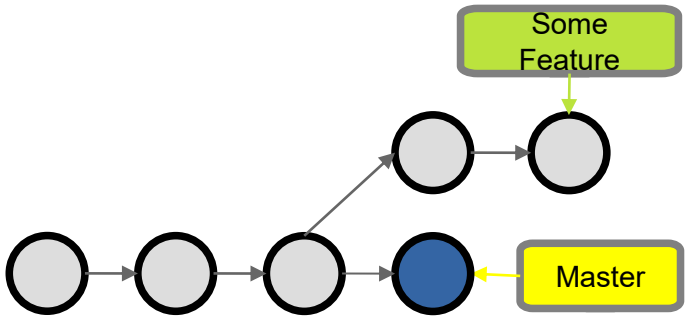


После Fast-Forward Merge

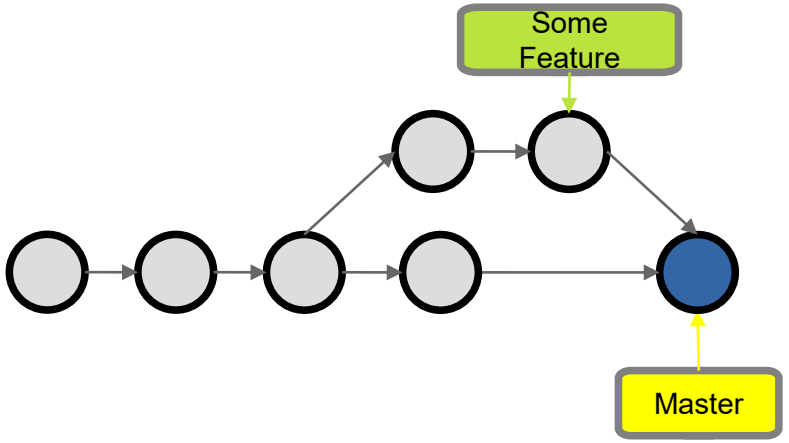


no-ff merge

До слияния



После слияния



Работа с ветками

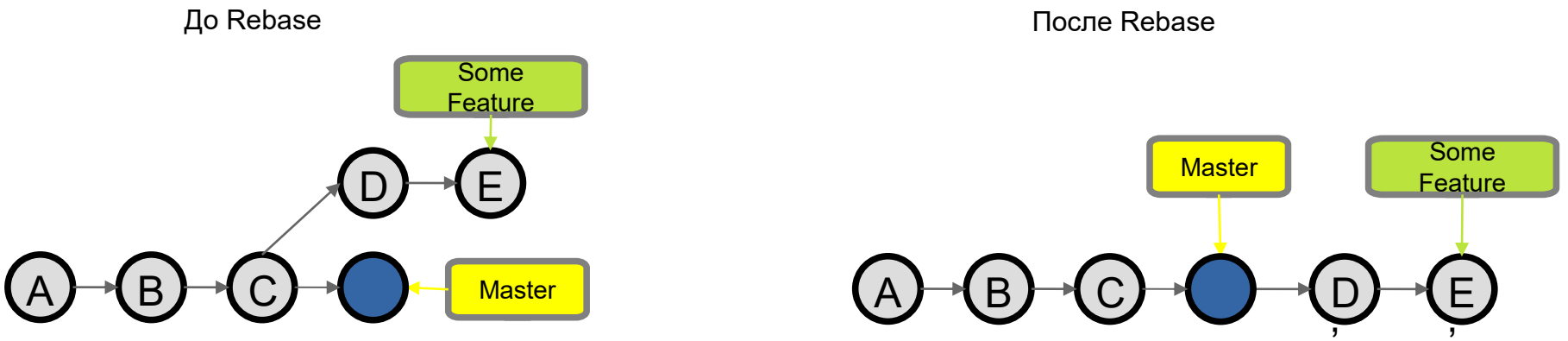
Rebase ветки

- git rebase <BRANCH> – rebase изменений в указанную ветку BRANCH
- git rebase -i <BRANCH> – интерактивный rebase, позволяющий редактировать, объединять или записывать коммиты
- git rebase --continue – продолжить rebase после разрешения конфликтов
- git rebase --skip – пропускает текущий patch и продолжает rebase
- git rebase --abort – Прерывает rebase и возвращается к исходному состоянию ветки до начала rebase

Работа с ветками

Преимущества использования rebase

- **Чистая история:** Rebase создает линейную историю коммитов, которую легче читать и понимать.
- **Упрощенное разрешение конфликтов:** Применяя изменения по одному, вы можете более систематически разрешать конфликты.
- **Лучшая интеграция:** Перебазирование позволяет плавно включать изменения из вышестоящего репозитория перед слиянием обратно в основную ветку.



Работа с удалёнными ветками

- git remote show <remote> – получить список удалённых веток
- <remote>/<branch> – имя отслеживаемой ветки
- git fetch <remote> – синхронизация локального репозитория с удалённым
- git push <remote> <BRANCH> - push изменений в удаленный репо на конкретную ветку
- git push <remote> <BRANCH_A>:<BRANCH_B> - push изменений с ветки BRANCH_A на удалённую ветку BRANCH_B
- git checkout -b <BRANCH> <remote>/<BRANCH> - pull изменений с удаленной новой ветки в локальную BRANCH
- git checkout --track <remote>/<BRANCH> - pull удаленной ветки в локальный репозиторий
- git pull – получение изменений с удаленной ветки в текущую локальную ветку
- git pull <remote> <BRANCH> - получить изменения с удаленной ветки BRANCH в локальную ветку

Регрессия

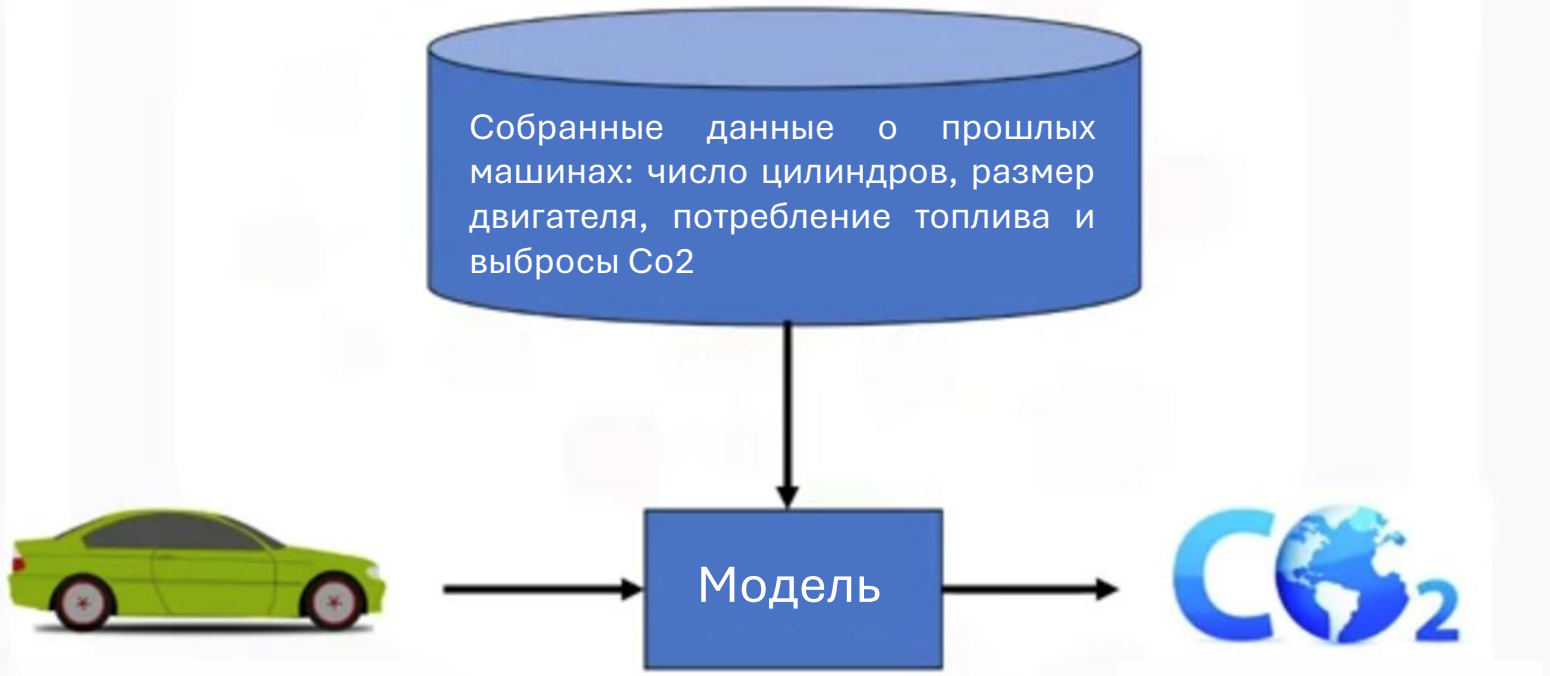
Введение

Simple Linear Regression

Практическое применение

Оценка модели и диагностика

Регрессия

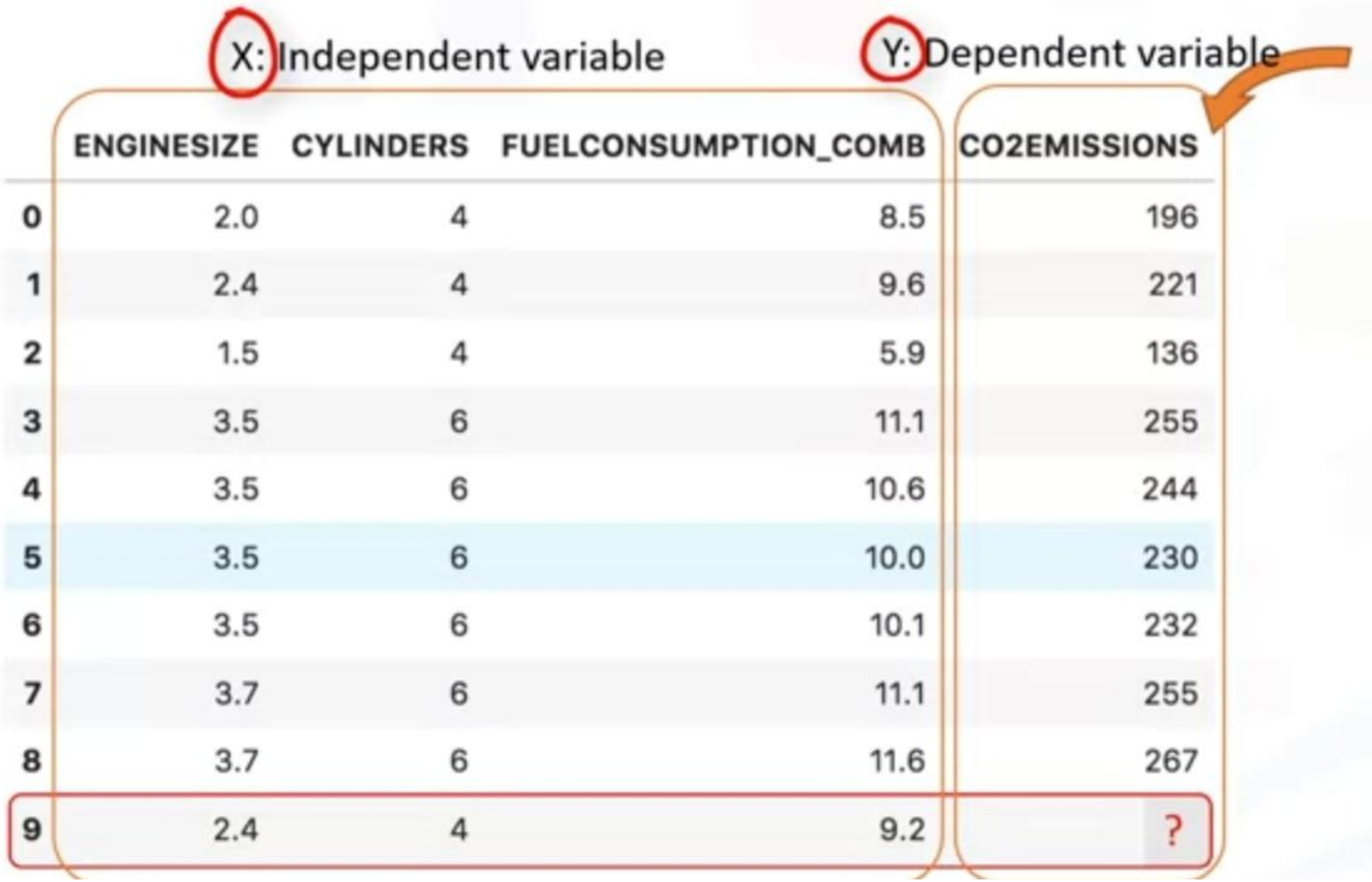

Собранные данные о прошлых машинах: число цилиндров, размер двигателя, потребление топлива и выбросы Co2
Модель
Новая машина
Ожидаемый выброс Co2

Регрессия

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Регрессия

- Задача предсказания непрерывного числа



	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Типы регрессии

Простая регрессия

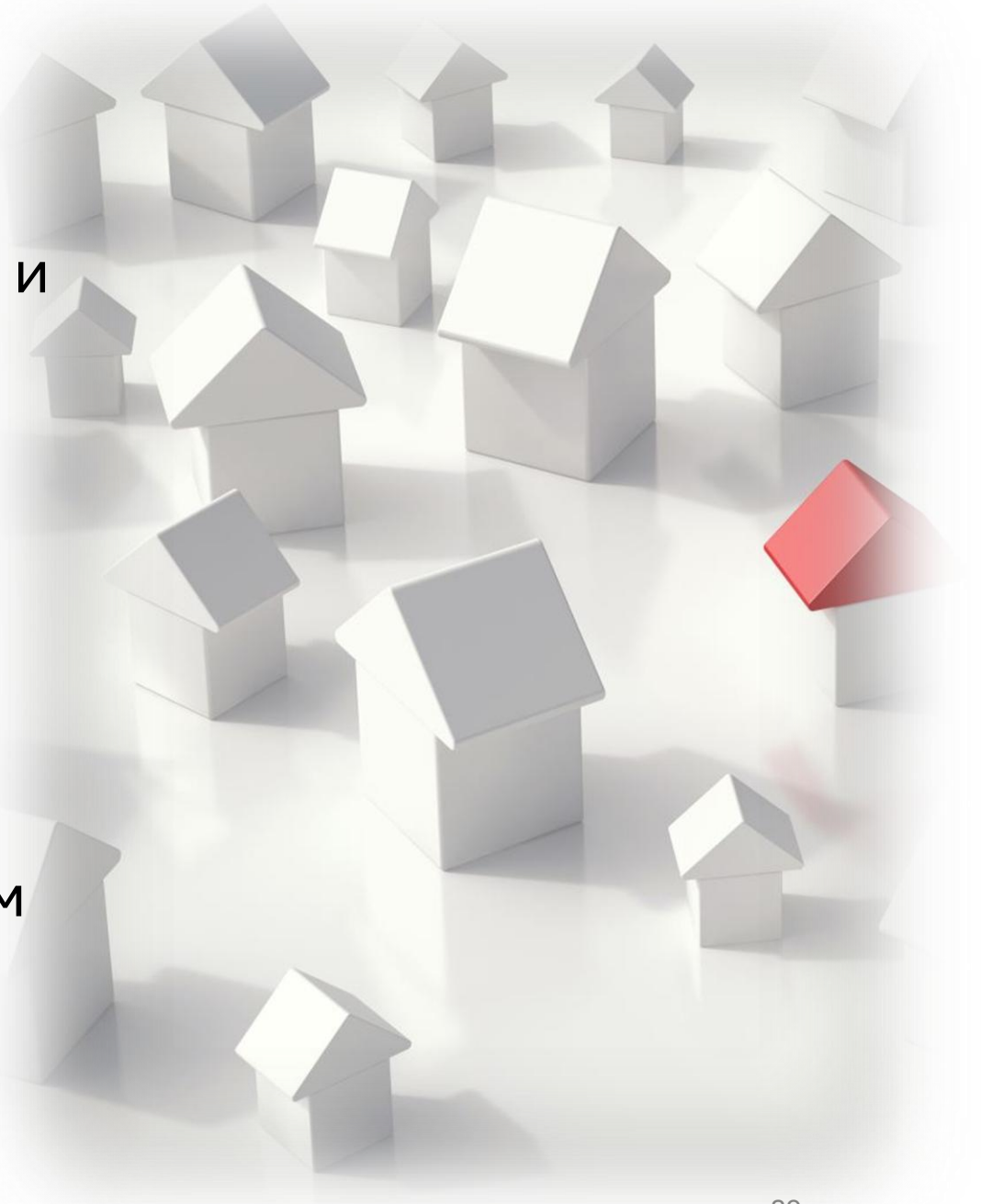
- Использование одной независимой переменной для прогнозирования зависимой переменной.
- Например: прогнозирование выбросов CO2 на основе размера двигателя.
- Простая линейная регрессия, простая нелинейная регрессия

Множественная регрессия

- Использование более чем одной независимой переменной для прогнозирования зависимой переменной.
- Например: прогнозирование выбросов CO2 на основе размера двигателя и количества цилиндров.
- Множественная линейная регрессия, множественная нелинейная регрессия

Применение регрессии

- Прогнозирование годового объема продаж человека на основе возраста, стажа работы и т. д
- Определение индивидуальной удовлетворенности на основе демографических и психологических факторов
- Прогнозирование цены дома на основе его размера, количества комнат и т. д
- Прогнозирование дохода от работы с учетом независимых переменных, таких как количество рабочих часов, образование, профессия, пол, возраст, стаж работы



Вопрос

Какой из примеров является примером применения регрессионного анализа?

- Прогнозирование наличия или отсутствия рака у пациента.
- Группировка похожих домов в районе.
- Прогнозирование количества осадков на следующий день.
- Прогнозирование победы или поражения команды.

Алгоритмы для решения задачи регрессии

- ❑Порядковая регрессия
- ❑Регрессия Пуассона
- ✓Линейная регрессия (Простая и множественная)
- ❑Полиномиальная регрессия
- ❑Регрессия Лассо
- ❑Гребневая регрессия
- ❑Регрессия на основе дерева решений
- ❑Регрессия на основе бустированного дерева решений

Простая линейная регрессия

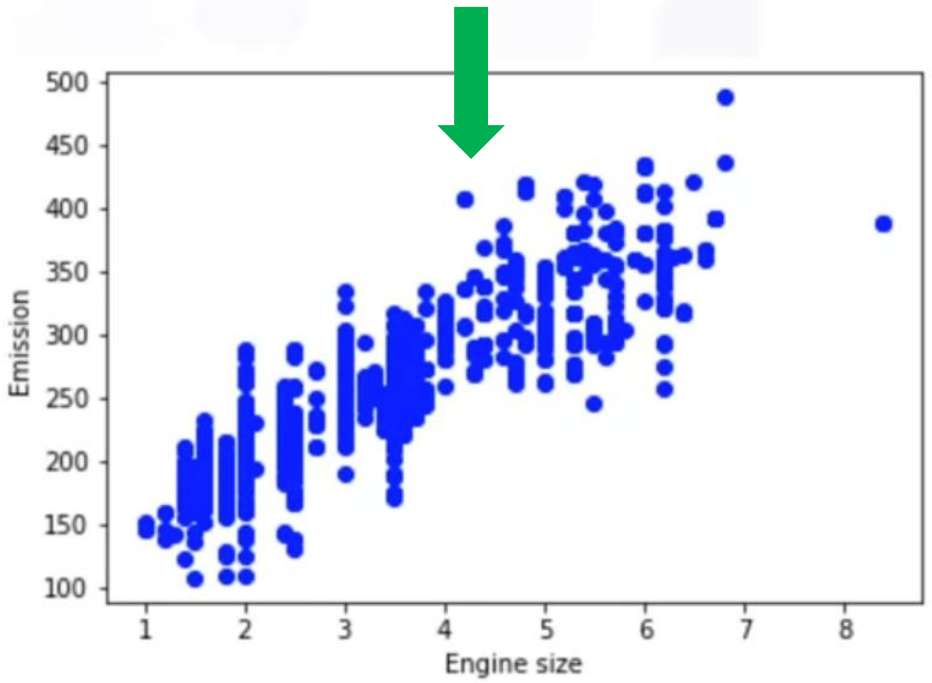
Линейная регрессия — это аппроксимация линейной модели, используемая для описания взаимосвязи между двумя или более переменными.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Простая линейная регрессия

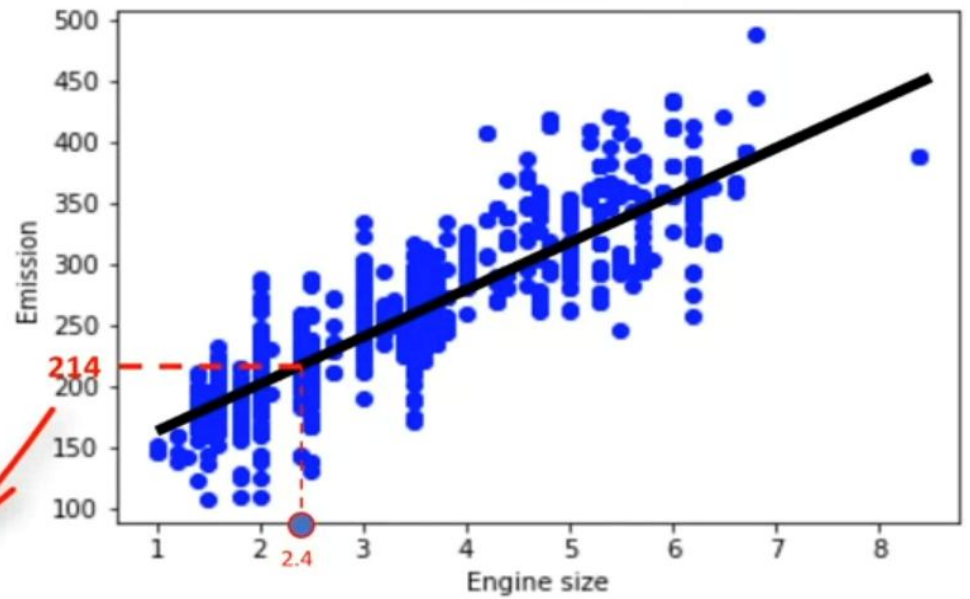
Изменение в одной переменной объясняет изменение в другой

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?



Простая линейная регрессия

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?



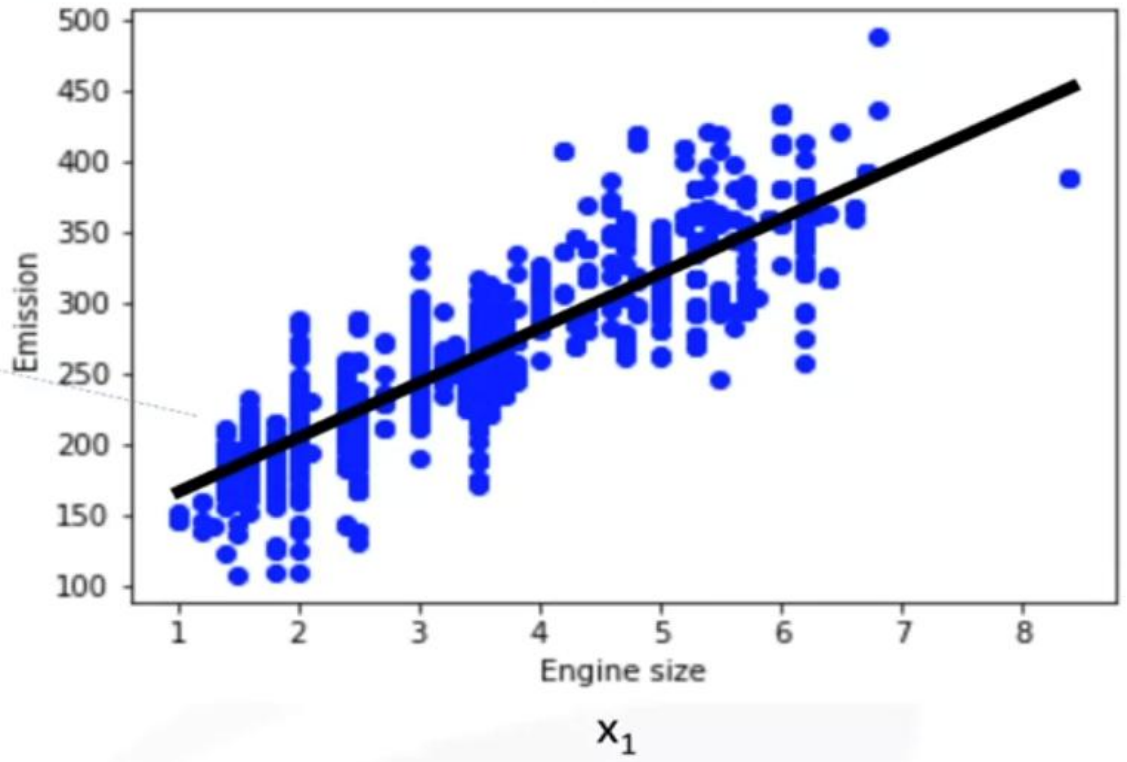
Простая линейная регрессия

Параметры, которые нужно найти

$$\hat{y} = \theta_0 + \theta_1 x_1$$

Независимая переменная

Зависимая переменная



Как найти оптимальный вариант?

$x_1 = 5.4$ независимая переменная

$y = 250$ истинный выброс Co2 для x1

$\hat{y} = \theta_0 + \theta_1 x_1$

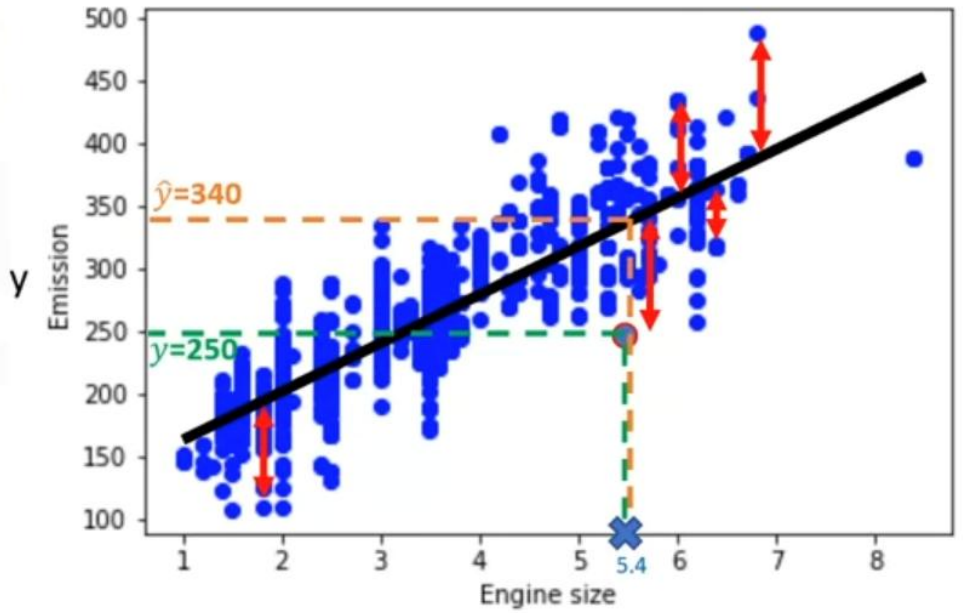
$\hat{y} = 340$ предсказанный выброс Co2 для x1

Error = $y - \hat{y}$

$= 250 - 340$

$= -90$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Математический подход

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

X_1 — ENGINESIZE; y — CO2EMISSIONS

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$$\theta_1 = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2}$$

$\bar{x}$ = (2.0 + 2.4 + 1.5 + ...)/9 = 3.0β

$\bar{y}$ = (196 + 221 + 136 + ...)/9 = 226.22

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

Минимизируя сумму квадратов ошибок (SSE)

Предсказание

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$$Co2Emission = \theta_0 + \theta_1 \, EngineSize$$

$$Co2Emission = 125 + 39 \, EngineSize$$

$$Co2Emission = 125 + 39 \times 2.4$$

$$Co2Emission = 218.6$$

Преимущества

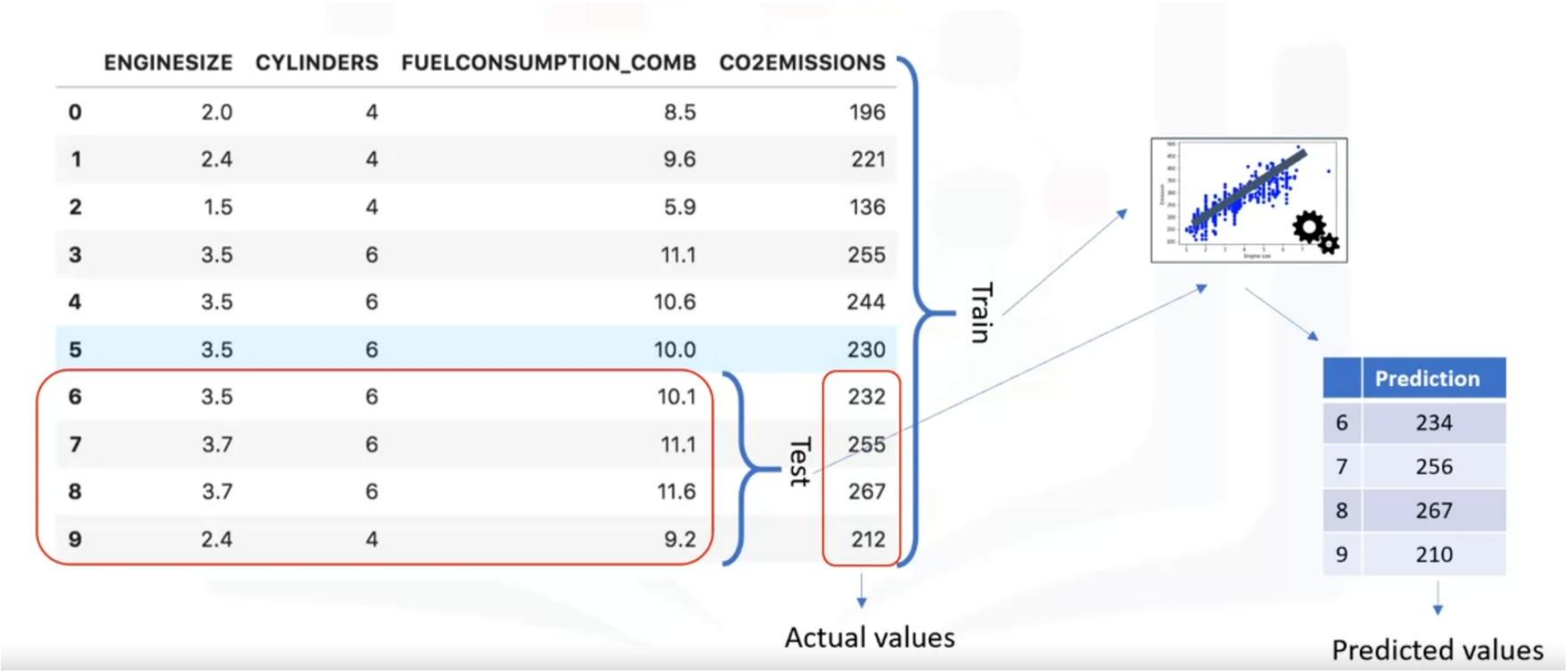
- Быстро
- Не нужна настройка гиперпараметров
- Интерпретируемая

Disadvantages

- Чувствительная к выбросам
- Не поможет при нелинейной связи

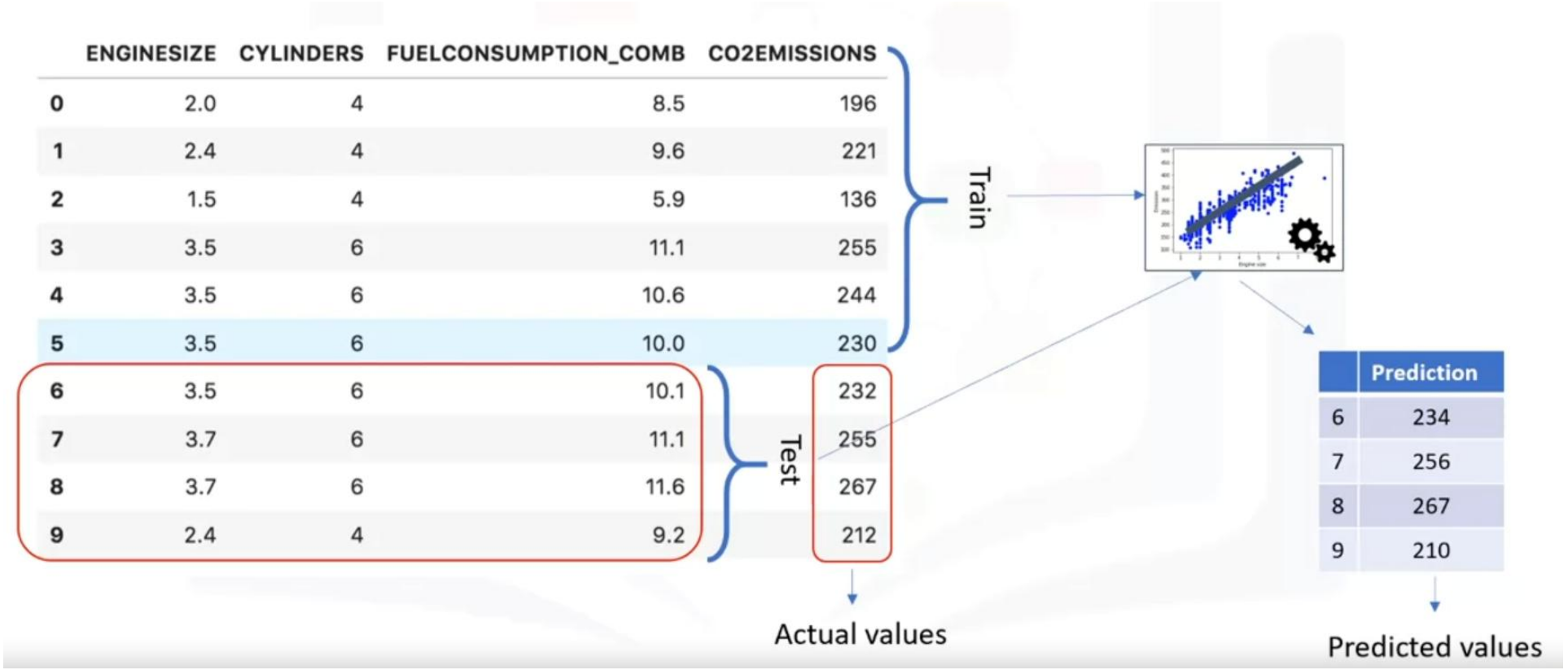
Как оценить точность (Accuracy) модели

Train/Test split



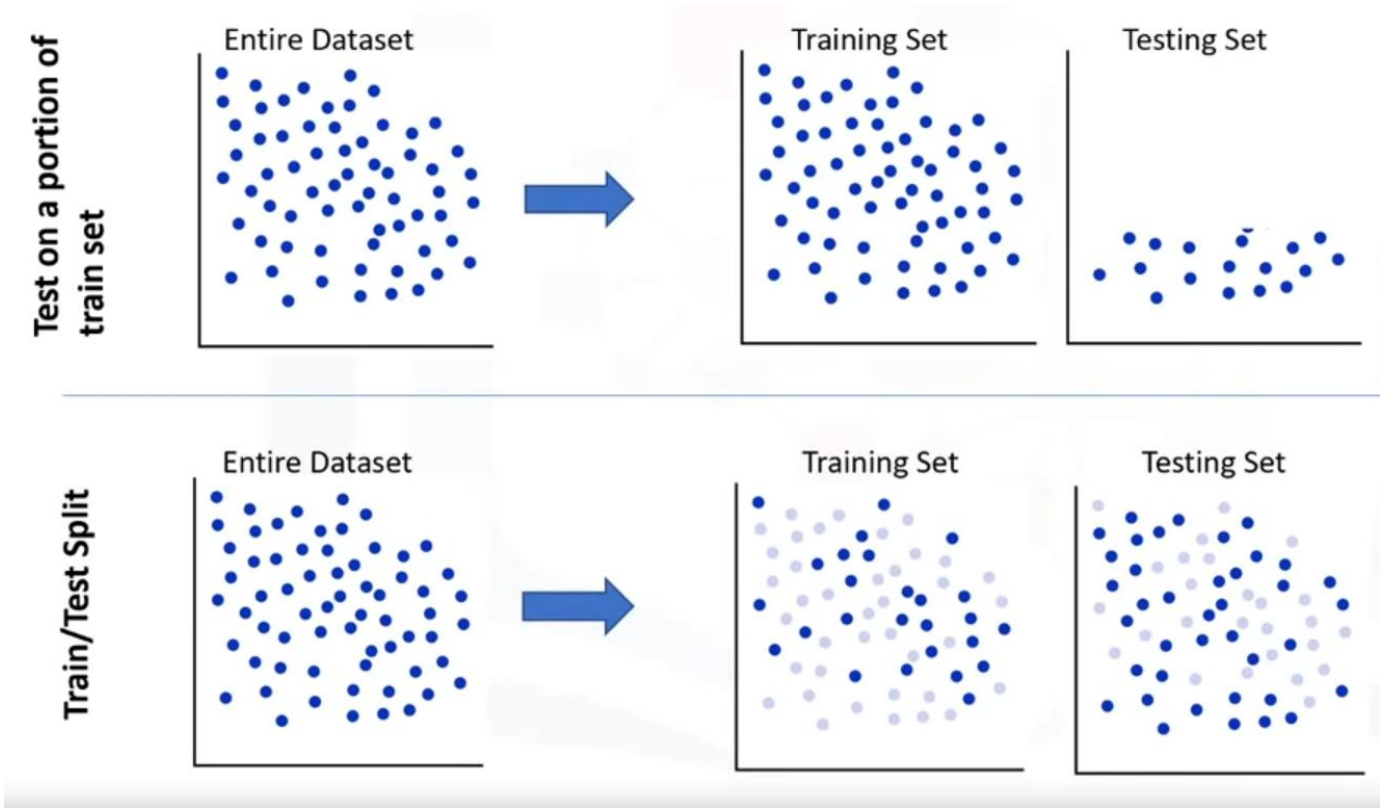
Как оценить точность (Accuracy) модели

Train/Test split



Как оценить точность (Accuracy) модели

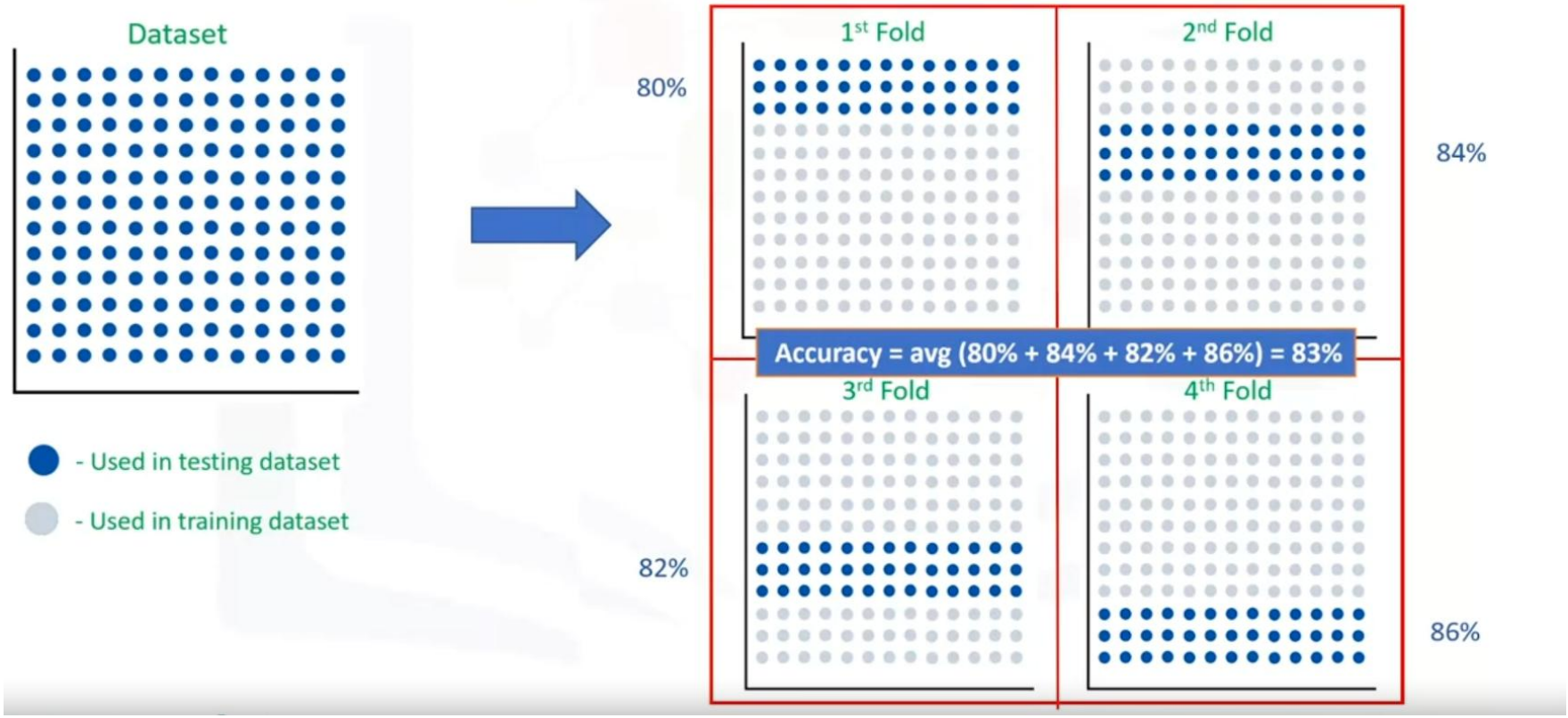
Train/Test split



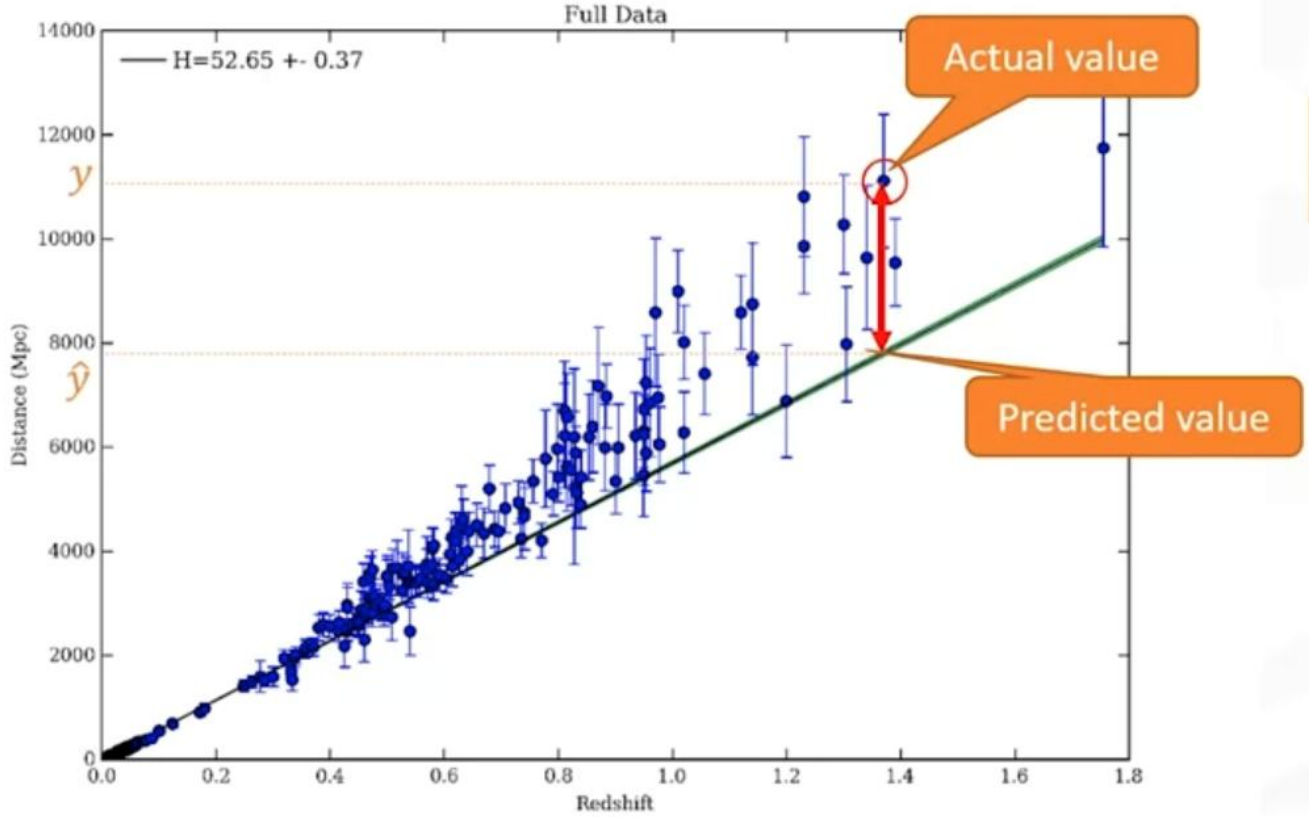
- Тестовая выборка часть обучающей
- Высокая точность на обученных данных
- Низкая точность на новых данных

- Разделение на независимые подвыборки
- Более высокая точность на новых данных
- Высоко зависит от того, на каких наборах данных обучалась и оценивалась

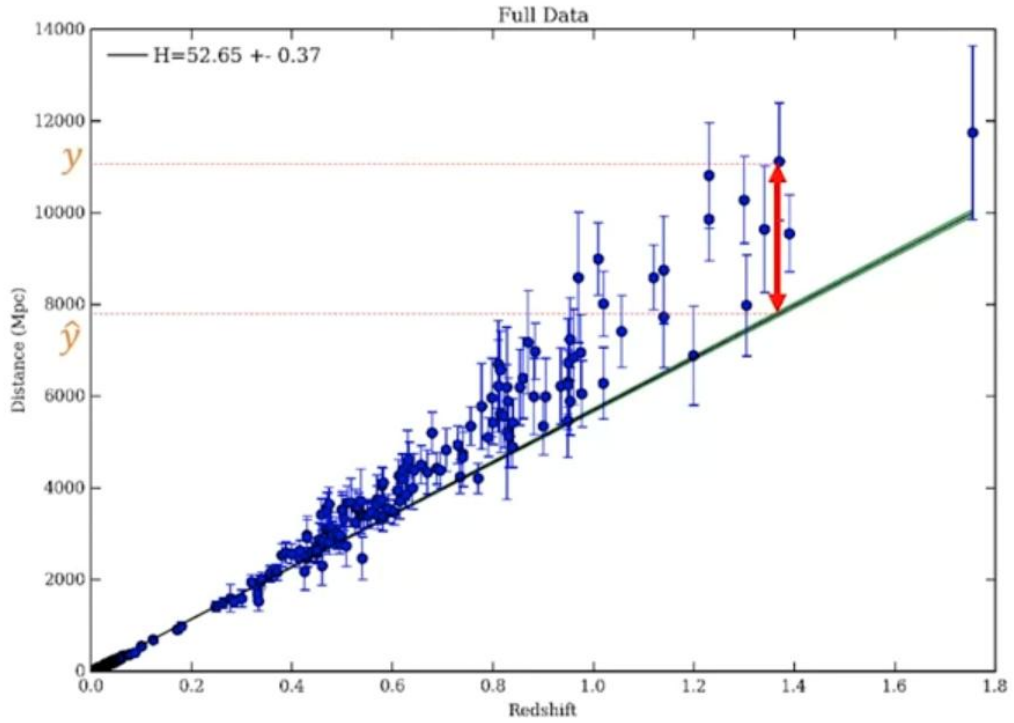
Кросс-валидация (k-fold cross-validation)



Оценка метрик в модели регрессии



Оценка метрик в модели регрессии



$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$RAE = \frac{\sum_{j=1}^{n}|y_j - \hat{y}_j|}{\sum_{j=1}^{n}|y_j - \bar{y}|}$$

$$RSE = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y})^2}$$

$$R^2 = 1 - RSE$$

Множественная линейная регрессия

$$Co2\ Em = \theta_0 + \theta_1 Engine\ size + \theta_2 Cylinders + \ ...$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ ... + \theta_n x_n$$

$$\hat{y} = \theta^T X$$

$$\theta^T = [\theta_0, \theta_1, \theta_2, ...] \qquad X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ ... \end{bmatrix}$$

X: Independent variable

Y: Dependent variable

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Как найти лучшие параметры?

$\hat{y} = \theta^T X$

$\hat{y}_i = 140$ предсказанный x_i

$y_i = 196$ истинный x_i

$y_i - \hat{y}_i = 196 - 140 = 56$ остаточная ошибка

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

Как найти лучшие параметры?

- Математический подход: операции линейной алгебры (для небольших наборов данных)
- Оптимизационный подход: градиентный спуск (для больших наборов данных)

Множественная линейная регрессия

$$Co2\ Em = \theta_0 + \theta_1 Engine\ size + \theta_2 Cylinders + \ ...$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ ... + \theta_n x_n$$

$$\hat{y} = \theta^T X$$

$$\theta^T = [\theta_0, \theta_1, \theta_2, ...] \qquad X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ ... \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y$$

X: Independent variable

Y: Dependent variable

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Множественная линейная регрессия

$$Co2\ Em = \theta_0 + \theta_1 Engine\ size + \theta_2 Cylinders + \ ...$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ ... + \theta_n x_n$$

$$\hat{y} = \theta^T X$$

$$\theta^T = [\theta_0, \theta_1, \theta_2, ...] \qquad X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ ... \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y$$

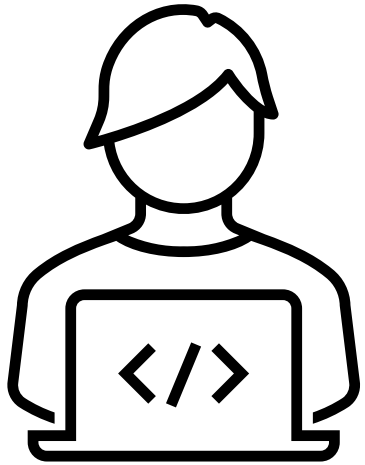
Смещение (Bias): Чтобы модель могла предсказывать значения не из начала координат, мы добавляем фиктивный столбец из единиц к матрице признаков.

Метрика R^2: Она показывает долю объясненной дисперсии. Значение 1.0 — идеальное предсказание, 0.0 — модель работает не лучше, чем простое предсказание среднего значения.

Ресурсы для изучения Git

- https://stepik.org/125248

Практика



Ресурсы для изучения ML

Лабораторные работы

Семинар 2:

https://colab.research.google.com/drive/1ClbOl4-sHVzBL8C4g9qEFI2OOIoWxsi-?usp=sharing

Лабораторная работа 2:

https://colab.research.google.com/drive/15x9TOZsFSvbrF-0bxKw8Wwk75N3voAM8?usp=sharing

Датасет для lab02:

https://www.kaggle.com/datasets/prokshitha/home-value-insights