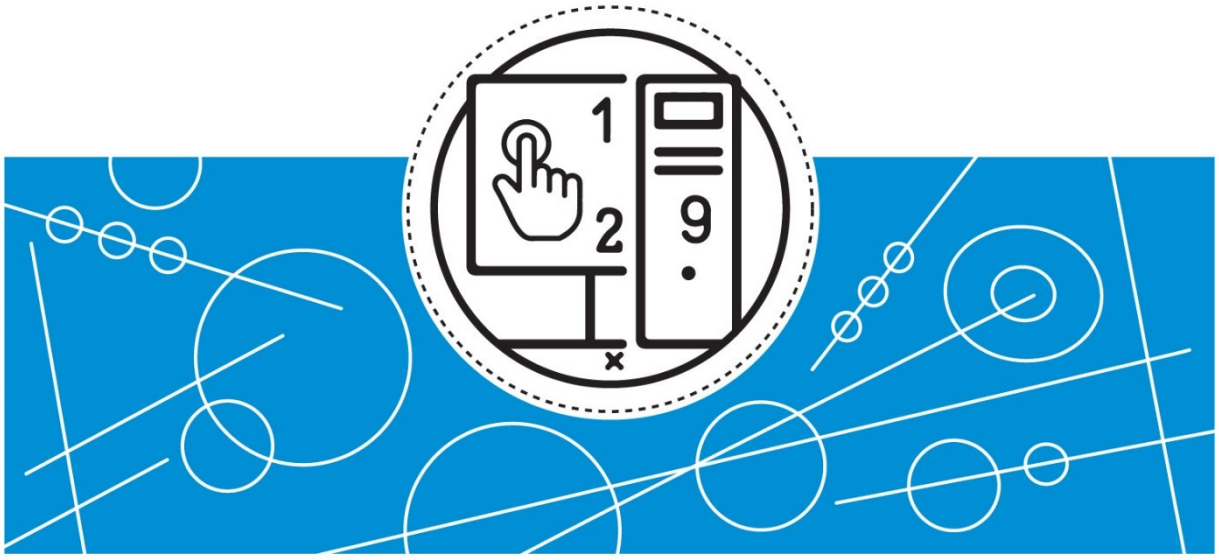
ИТМО

О. А. ЕВСТАФЬЕВ, С. А. КИМ

ПРИКЛАДНОЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ ЛАБОРАТОРНЫЙ ПРАКТИКУМ



Санкт-Петербург
2026

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

О. А. Евстафьев, С. А. Ким

ПРИКЛАДНОЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ
В УНИВЕРСИТЕТЕ ИТМО
для студентов бакалавриата образовательной программы
«Робототехника и искусственный интеллект»
в качестве практикума для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2026

УДК 004.8(076.5)
ББК 16.63я73
Е26
О. А. Евстафьев, С. А. Ким. Прикладной искусственный интеллект. Лабораторный практикум. – Санкт-Петербург: Университет ИТМО, 2026. – 60 с.

Рецензент:
Шаветов Сергей Васильевич, кандидат технических наук, доцент факультета систем управления и робототехники, Университет ИТМО, Санкт-Петербург, Россия.

Лабораторный практикум по дисциплине «Прикладной искусственный интеллект» посвящен основам машинного обучения и нейронных сетей. Представлены три лабораторные работы: классификация активности человека по данным мобильных сенсоров, прогнозирование стоимости жилья с помощью нейросетевой регрессионной модели и распознавание видов цветов с использованием сверточных нейронных сетей и переноса обучения.

Каждая работа включает теоретические сведения, порядок выполнения, примеры вычислений, планы экспериментов и контрольные вопросы. Рассматриваются подготовка данных, разделение выборок, выбор метрик, настройка моделей, анализ ошибок и обеспечение воспроизводимости результатов. Электронное приложение содержит ноутбуки Jupyter и вспомогательные модули на Python.

Практикум предназначен для студентов бакалавриата образовательной программы «Робототехника и искусственный интеллект», по направлениям: 12.03.01 «Приборостроение»; 13.03.02 «Электроэнергетика и электротехника»; 15.03.04 «Автоматизация технологических процессов и производств»; 15.03.06 «Мехатроника и робототехника»; 24.03.02 «Системы управления движением и навигация»; 27.03.04 «Управление в технических системах» (факультет систем управления и робототехники Университета ИТМО). Предполагается владение основами Python; предварительные знания машинного обучения не требуются.

Университет ИТМО (Санкт-Петербург) – национальный исследовательский университет и научно-образовательная корпорация. Альма-матер победителей международных соревнований по программированию. Приоритетные направления: IT и искусственный интеллект, фотоника, робототехника, квантовые коммуникации, трансляционная медицина, науки о жизни, Art&Science и научная коммуникация.

ИТМО – участник федеральной программы «Приоритет-2030». Университет реализует программу развития на 2025–2030 годы, ориентированную на сотрудничество с индустрией. С 2022 года ИТМО работает по новой модели развития – научно-образовательная корпорация: академическая свобода, поддержка инициатив студентов и сотрудников, распределенная система управления, приверженность открытому исходному коду и бизнес-подходы к организации работы. Обучение в университете строится вокруг индивидуальной образовательной траектории каждого студента.

Университет представлен в международных предметных рейтингах, в том числе входит в топ-150 мировых университетов в области Automation & Control (кибернетика) согласно Shanghai Ranking. По данным SuperJob за 2026 год, ИТМО занимает первое место в Санкт-Петербурге и второе в России по уровню зарплат выпускников в IT-сфере.

© Университет ИТМО, 2026
© О. А. Евстафьев, С. А. Ким, 2026

Содержание

Введение 4
Первое знакомство с машинным обучением 5
Подготовка среды и исходных материалов 11
Практическая работа № 1. Классификация 13
1.1 Задача распознавания активности 13
1.2 От сигналов к признакам 13
1.3 Разделение данных по участникам 14
1.4 Три семейства классификаторов 15
1.5 Метрики и матрица ошибок 16
1.6 Порядок выполнения 17
1.7 Отчет и вопросы защиты 20
Практическая работа № 2. Нейронные сети 22
2.1 Постановка задачи 22
2.2 Как используются исходные таблицы 22
2.3 Пропуски, категории и масштабирование 22
2.4 Полносвязная нейронная сеть 24
2.5 Функции активации 25
2.6 Функции потерь и метрики регрессии 26
2.7 Обучение: градиент, эпоха и batch 27
2.8 Переобучение и воспроизводимость 28
2.9 Порядок выполнения 29
2.10 Выполнение на PyTorch 33
2.11 Исследование гиперпараметров 34
2.12 Разбор ошибок при выполнении работы 37
2.13 Отчет и защита работы № 2 38
Практическая работа № 3. Сверточные нейронные сети 40
3.1 Постановка задачи 40
3.2 Изображение как тензор 40
3.3 Сверточный слой 41
3.4 Пулинг и устройство исходной сети 42
3.5 Логиты, перекрестная энтропия и обучение 43
3.6 Перенос обучения и заморозка 43
3.7 Подготовка изображений и разбиение 44
3.8 Обучение и исследование архитектур 45
3.9 Сравнение, визуализация и сохранение 47
3.10 Контрольные вопросы 49
Общие требования к отчету 50
Литература 51
Приложение А. Материалы эксперимента 52
Приложение Б. Шаблон отчета 54
Приложение В. Глоссарий 55

Введение

Построение правил распознавания и прогнозирования по примерам называют **машинным обучением (англ. machine learning, ML)**. Это одно из направлений искусственного интеллекта, которому посвящены три работы по дисциплине «Прикладной искусственный интеллект».

В каждой работе предстоит выбрать модель и объяснить ее ошибки. Сначала нужно определить, что известно об объекте к моменту прогноза и как проверить ответ. От этого зависят состав входных данных, способ разделения примеров и критерий сравнения.

Таблица 1: Содержание лабораторных работ

№	Задача	Что исследуется	Материалы отчета
1	Классификация активности	Сравнение алгоритмов; разделение по участникам	Отчет по шести классам и матрица ошибок
2	Регрессия стоимости	Подготовка смешанных признаков; обучение полносвязной сети	Ошибки прогнозов и таблица цен
3	Классификация цветов	Свертки, перенос обучения, управление режимами модели	Сравнение трех архитектур на одной валидации

Входная подготовка

Нужны функции и коллекции Python, чтение таблиц в pandas, простые операции с массивами NumPy, среднее и представление о производной. Вектор можно сначала понимать как список чисел, матрицу – как прямоугольную таблицу. Основы машинного обучения изложены в следующем разделе.

После вводных разделов выполняйте работы по порядку.

Работа с примерами

Учебный модуль – файл Python с функциями, которые вызываются из ноутбука. Перед выполнением ячейки определите ее вход и ожидаемый результат; после выполнения проверьте, совпало ли ожидание с наблюдением. Самостоятельная часть работы состоит в выборе настроек и объяснении результатов сравнения.

Для дальнейшего чтения о нейронных сетях можно использовать книгу «Глубокое обучение» [1], а о разработке и эксплуатации систем машинного обучения – Machine Learning System Design [2]. Английские эквиваленты приведены при введении понятий; краткий словарь со ссылками на объяснения находится в приложении В.

Первое знакомство с машинным обучением

Когда правило приходится искать по данным

Представим, что нужно оценить стоимость дома до его продажи. Площадь, расположение и состояние дома уже известны, а цена будущей сделки – еще нет. Можно начать с простого рассуждения: при прочих равных больший дом обычно стоит дороже. Но сколько добавляет каждый квадратный метр? Насколько важны возраст здания и район? Записать правдоподобную зависимость нетрудно; подобрать ее так, чтобы она давала полезные прогнозы, гораздо сложнее.

Здесь помогают сведения о прошлых продажах. Для них известны и характеристики домов, и фактические цены. Такие примеры называют **обучающими данными (англ. training data)**, а настраиваемое правило прогноза – **моделью (англ. model)**. Алгоритм обучения использует примеры, чтобы выбрать значения внутри этого правила. Затем обученную модель можно применить к дому, которого в исходной таблице не было.

Прежде чем выбирать алгоритм, уточним, что будет одной строкой нашей таблицы. В примере это один проданный дом – **объект (англ. sample)**. Его известные характеристики называются **признаками (англ. features)**; цена, которую требуется предсказать, – **целевым ответом (англ. target)**. Совокупность объектов образует **набор данных (англ. dataset)**. Выделенную для определенной цели часть набора будем называть выборкой.

Это описание уже задает границы задачи. Модель получает только записанные признаки. Если в таблице нет сведений о состоянии дома, она не сможет непосредственно учесть аварийную крышу. Если вместо окончательной цены сделки записана цена объявления, модель будет учиться предсказывать именно ее. Поэтому содержательный смысл столбцов нужно выяснить до запуска обучения.

Предсказание цены относится к **регрессии (англ. regression)**: ответом служит число. В **классификации (англ. classification)** нужно выбрать одну из заданных категорий. Например, по показаниям датчиков телефона определить, идет человек или сидит. Категории называют классами, а правильный класс объекта – его **меткой (англ. label)**. Даже если классы закодированы числами, расстояние между этими кодами обычно не имеет смысла: код 4 не означает «вдвое больше активности», чем код 2.

В обеих задачах обучающие примеры сопровождаются правильными ответами. Такую постановку называют **обучением с учителем (англ. supervised learning)**. «Учитель» здесь – известные ответы в данных: цены состоявшихся сделок, названия видов цветов, записи о выполненных человеком движениях. Они могут поступать из измерений, документов или работы разметчиков. Ошибки в этих ответах тоже становятся частью данных, по которым обучается модель. Именно обучение с учителем рассматривается во всех трех работах практикума.

Что именно изменяется при обучении

Чтобы увидеть обучение в самом простом виде, оставим у дома один признак – площадь. Возьмем три вымышленные продажи; цены указаны в миллионах

рублей. Этих трех строк достаточно для ручного расчета, но для оценки качества реальной модели понадобилось бы значительно больше наблюдений.

Дом	1	2	3
Площадь, м^2	50	80	110
Цена, млн руб.	6	9	10

Предположим, что прогноз цены можно получить по правилу

$$\widehat{y} = as + b,$$

где s – площадь, a – прибавка к прогнозу на единицу площади, а b – постоянное слагаемое. Знак «шляпка» отличает прогноз $\widehat{y}$ от известной цены y. Числа a и b – **параметры (англ. parameters)** этой модели. На графике изменение a поворачивает прямую, а изменение b сдвигает ее вверх или вниз.

Сравним два кандидата. У первого $a = 0{,}10$, $b = 0$, поэтому прогнозы равны 5, 8 и 11. У второго $a = 0{,}08$, $b = 2$, и получаются 6, 8,4 и 10,8. Какой вариант считать лучше? Для ответа нужно заранее договориться, как измерять ошибку. Такое численное правило оценки называют **метрикой качества (англ. metric)**.

Для каждого дома возьмем модуль разности известной цены и прогноза: завышение и занижение на одну сумму дадут одинаковую ошибку. Среднее этих ошибок называется **средней абсолютной ошибкой (англ. mean absolute error, MAE)**. Для первого правила, $\widehat{y} = 0{,}10s$, получим

$$\mathrm{MAE}_1 = \frac{|6-5| + |9-8| + |10-11|}{3} = \frac{1+1+1}{3} = 1 \text{ млн руб.}$$

Для второго правила, $\widehat{y} = 0{,}08s + 2$,

$$\mathrm{MAE}_2 = \frac{|6-6| + |9-8{,}4| + |10-10{,}8|}{3} = \frac{0+0{,}6+0{,}8}{3} \approx 0{,}47 \text{ млн руб.}$$

Чем меньше MAE, тем меньше средняя абсолютная ошибка. Поэтому из двух сравниваемых правил $\widehat{y} = 0{,}08s + 2$ лучше описывает эти три продажи: $0{,}47 < 1$. На рисунке 1 показаны оба ненулевых отклонения второго правила. Для дома площадью 50 м^2 прогноз совпал с ценой.

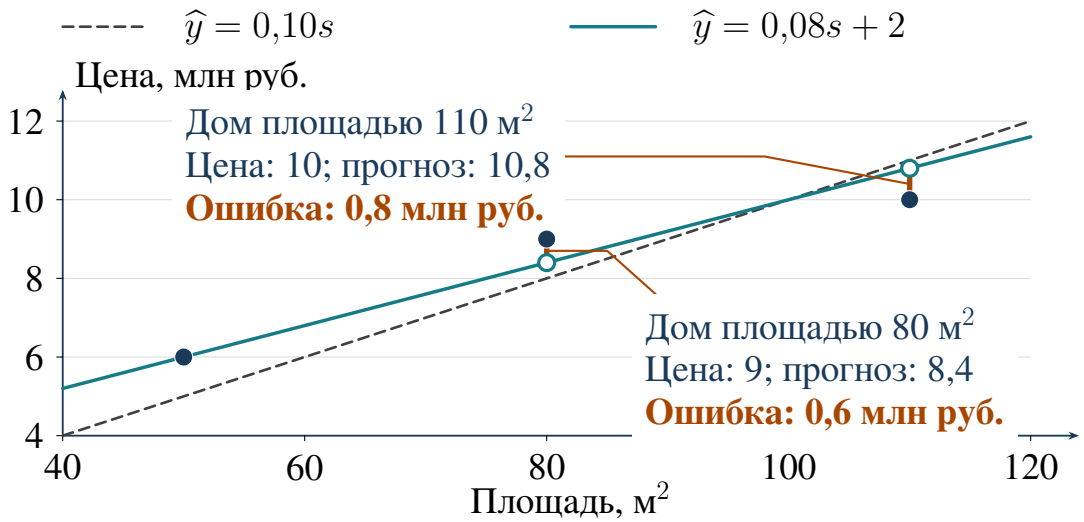


Рис. 1: Два правила прогноза на вымышленных данных. Закрашенные точки – цены, пустые – прогнозы второго правила; выделенные отрезки – его абсолютные ошибки.

Мы пока только сравнили два вручную заданных варианта. Алгоритм обучения проделывает систематический поиск подходящих параметров. Способ поиска зависит от модели и выбранного критерия: где-то параметры можно вычислить сразу, где-то приходится постепенно улучшать текущее решение. Для нейронной сети во второй работе будет показано, как вычисленная ошибка подсказывает направление изменения ее коэффициентов – весов. Поэтому слово «обучение» обозначает вполне конкретную вычислительную процедуру.

Полезно различать выбор формы правила и настройку чисел внутри него. Решение учитывать только площадь или добавить другие признаки принимает исследователь. Можно использовать и другое правило: оценивать цену по одному наиболее похожему проданному дому или по нескольким. Число таких соседей тоже задает исследователь. Подобные настройки называются **гиперпараметрами (англ. hyperparameters)**. Выбранная конфигурация задает условия обучения. Сравнивать варианты по новым объектам мы научимся ниже.

Как читать формулы и вызовы в ноутбуке

Когда признаков несколько, удобно собрать их в вектор – упорядоченный список чисел. Например, $x_i = (80, 3)$ может означать площадь и число комнат дома с номером i. Порядок существенен: перестановка столбцов меняет смысл входа. Буквой y_i обозначим правильный ответ для этого дома, а $\widehat{y}_i$ – прогноз. Общее обозначение модели – f_θ, где θ объединяет ее настраиваемые параметры:

$$\widehat{y}_i = f_\theta(x_i).$$

В нашем примере с одной площадью эта запись скрывает знакомое действие: умножить площадь на найденное a и прибавить найденное b. У нейросети вычислений будет больше, но назначение входа, параметров и результата останется тем же.

Признаки всех домов можно записать строками одной матрицы X. Тогда обучение получает и X, и соответствующие ответы y. В scikit-learn этому обычно соответствует вызов `model.fit(X_train, y_train)`. После обучения для получения прогнозов используется `model.predict(X_new)`: правильных ответов для новых домов в этом вызове нет. **Применение обученной модели, или инференс (англ. inference)**, использует уже найденные параметры. Сам по себе вызов `predict` их не переобучает.

Иногда до подачи в модель значения нужно подготовить: заполнить пропуски, перевести категории в числа, изменить масштаб. Эти операции называют **предобработкой (англ. preprocessing)**. Часть ее правил тоже определяется из данных. Например, число для заполнения пропусков можно вычислить по известным значениям обучающих домов. При применении модели сохраняется вся согласованная последовательность: сначала те же преобразования, затем прогноз.

Почему маленькой ошибки на обучении недостаточно

У нас получилось уменьшить ошибку на трех известных продажах. Но покупателю нужна оценка другого дома. Если его площадь равна 90 м^2, второе правило даст 9,2 млн руб. Проверить этот ответ можно только после того, как станет известна настоящая цена. Из хорошего совпадения с тремя прежними продажами его точность еще не следует.

Можно добиться и нулевой ошибки на известных объектах: сохранить их характеристики вместе с ценами и возвращать записанный ответ при совпадении. Для дома, которого нет в этой таблице, все равно придется выбрать способ прогноза. Этот пример показывает, почему воспроизвести знакомые ответы недостаточно. Нас интересует **обобщение (англ. generalization)** – способность модели давать полезные ответы на новых объектах.

Более гибкая модель может учесть особенности, которые случайно встретились именно в обучающих данных. Например, дорогие дома могли попасть в маленькую выборку преимущественно с одним цветом фасада. Такая зависимость поможет описать исходные строки, но может не повториться в следующих продажах. Излишнюю подстройку под особенности обучения называют **переобучением (англ. overfitting)**. Его признак – заметное расхождение качества на обучающих и новых сопоставимых объектах. Одного низкого значения ошибки обучения для вывода о качестве недостаточно.

Возможна и обратная проблема: правило слишком грубо, чтобы описать доступную зависимость. Модель, которая назначает всем домам одну цену, совсем не реагирует на площадь. Если полезная закономерность не освоена и ошибки велики и на обучении, и на проверке, говорят о **недообучении (англ. underfitting)**. Причиной может быть как ограниченная модель, так и неудачная настройка ее обучения. По двум числам ошибки причину однозначно не определить: нужны дополнительные опыты.

При этом постоянный прогноз полезен как **базовая модель (англ. baseline)** – исходный вариант для сравнения. Если более сложное решение не превосходит такой ориентир на отложенных данных, польза его усложнения пока не показана.

В лабораторной нужно выяснить причину: например, признаки мало связаны с ответом, неверно подготовлены или модель не успела обучиться.

Какие данные нужны для проверки

Чтобы сравнение было содержательным, заранее отложим часть объектов. **Обучающая выборка (англ. training set, train)** нужна для настройки параметров: по ее домам алгоритм подбирает a и b, а по ее значениям обучаются правила предобработки. Известные ответы **валидационной выборки (англ. validation set)** используются для сравнения вариантов. По ней можно решить, помогают ли добавленные признаки или другой набор гиперпараметров. Это сравнение и выбор называют валидацией.

Валидация тоже постепенно влияет на решение: мы видим, какой вариант выиграл, и планируем следующий опыт с учетом этого результата. Если перебирать варианты достаточно долго, можно подстроиться и под особенности валидационных объектов. Для итоговой оценки выбранного решения нужен **тест (англ. test set)** – еще одна часть с известными ответами, которая не участвовала ни в обучении, ни в выборе. После просмотра теста и изменения модели прежнюю оценку уже нельзя считать независимой проверкой нового решения.

Важно, какие именно объекты отложены. Если мы хотим распознавать движения нового человека, проверочные записи должны принадлежать людям, которых не было в обучении. Соседние фрагменты записи одного участника могут быть очень похожи: их случайное распределение по двум частям не проверяет перенос на другого человека. В первой работе мы вернемся к этому вопросу, когда будем делить данные по идентификаторам участников.

Есть и менее заметный способ испортить проверку. Предположим, в признаки добавили цену квадратного метра, вычисленную из окончательной цены сделки. Она помогает восстановить целевой ответ, но до продажи ее нет. Возникает **утечка данных (англ. data leakage)**: обучение использует информацию, недоступную в условиях будущего применения. Утечку создает и вычисление статистик предобработки по всему набору до его разделения: подготовленные обучающие данные тогда уже зависят от отложенных объектов. Поэтому сначала выделяют выборки, а обучаемые преобразования настраивают только на обучающей части.

В практикуме названия файлов отражают разные соглашения. В работе № 1 `test.csv` содержит правильные ответы для итоговой проверки. В работе № 2 одноименный файл содержит новые дома без цен: для него можно сформировать прогнозы, но нельзя вычислить их ошибку. Валидацию выделяют из размеченной таблицы. В работах № 2 и № 3 сравнение на валидации характеризует выбор модели в условиях учебного опыта; для независимой итоговой оценки понадобились бы дополнительные размеченные данные.

Даже аккуратно выполненное разбиение не обещает хорошей работы в любых условиях. Дома другого региона, новый способ измерения или фотографии при другом освещении могут заметно отличаться от изученных примеров. Результат проверки следует связывать с тем, откуда получены данные и для каких объектов предполагается применять модель.

Как превратить запуск программы в эксперимент

В первой работе готовые функции помогут загрузить данные и обучить классификаторы. Содержательная часть начинается с предположения, которое можно проверить: например, поможет ли изменение числа соседей лучше распознавать движения новых людей. Для сравнения понадобится один и тот же набор проверочных участников. Если одновременно заменить и модель, и разбиение, будет трудно понять, чем вызвана разница результатов.

Порядок опыта называют **протоколом**: какие данные используются, как они разделены, что меняется и какая метрика сравнивается. **Бюджет** ограничивает число запусков, проходов по данным или время. Эти условия стоит записать до подбора модели: тогда неудачный запуск тоже дает сведения о проверяемом предположении.

Некоторые операции используют случайность: например, выбор части объектов или начальных весов сети. **Начальное значение генератора случайных чисел (англ. random seed)**, далее seed, помогает повторить такую последовательность действий. Вместе с ним сохраняют разбиение, настройки и версии библиотек. Повторяемость запуска позволяет разобрать результат, а его устойчивость выясняют отдельными опытами: совпадение одного запуска само по себе не доказывает надежность модели.

> **Небольшая проверка перед первой работой**
>
> Пусть модель по истории продаж предсказывает цену нового дома.
>
> 1. Можно ли включить в признаки стоимость квадратного метра, рассчитанную по окончательной цене этой же сделки?
> 2. Модель безошибочно воспроизвела все обучающие цены. Достаточно ли этого, чтобы доверять ее оценке нового дома?
> 3. Два варианта сравнили на разных наборах домов. Что нужно изменить в опыте, чтобы сравнение стало убедительнее?
>
> Ориентиры для ответа: момент доступности признака; проверка обобщения; одни и те же отложенные объекты и метрика.

Более подробное введение в постановки задач и основы моделей можно найти в книге С. И. Николенко «Машинное обучение: основы» [3]. Здесь изложенных понятий достаточно, чтобы перейти к данным первой работы и понимать назначение каждого этапа эксперимента.

Подготовка среды и исходных материалов

Электронное приложение к практикуму:
`https://github.com/stalkim/itmo-ml-practicum`

Структура файлов

Скачайте весь проект: ноутбуки используют функции из общей папки `code`. Данные и результаты каждой работы находятся в ее собственном каталоге. Файлы CSV (англ. comma-separated values) хранят текстовую таблицу со строками и разделителями столбцов; в учебных файлах разделителем служит запятая.

```
project/
  code/                          # common teaching modules
  labs/
    01_classification/
      lab1_activity.ipynb
      data/                      # train.csv, test.csv
    02_regression/
      lab2_housing.ipynb
      data/                      # CSV and data_description.
   txt
    03_cnn/
      lab3_flowers.ipynb
      data/                      # five class folders
```

Работа	Что должно быть в данных
№ 1: активность	В обеих таблицах: числовые признаки, метка `Activity`, идентификатор участника (в коде – `subject`).
№ 2: жилье	В обучающей таблице: `Id`, признаки и `SalePrice`; в таблице прогнозирования цены нет. Описание признаков находится в `data_description.txt`.
№ 3: цветы	Пять каталогов видов с изображениями. Индексы классов определяются фактическим отображением `class_to_idx`.

Файлы `train.csv` двух работ относятся к разным задачам. В репозитории размещены таблицы активности и жилья; недостающие материалы получите у преподавателя. Для третьей работы используется набор Flowers Recognition [4]. Размеры выборок определяйте после чтения своей версии файлов.

Что такое ноутбук и как его запускать

Ноутбук Jupyter (англ. notebook) – файл `.ipynb`, который состоит из текстовых ячеек и ячеек Python. Выполняйте их сверху вниз: переменные сохраняются в памяти между запусками. Если после изменения данных повторить только нижние ячейки, часть расчетов может опираться на старые значения. Код

исполняет **ядро Jupyter (англ. kernel)** – отдельный процесс, который хранит переменные. Перед сдачей перезапустите ядро и выполните ноутбук целиком.

Запускайте код в Jupyter Notebook или JupyterLab на компьютере либо учебном сервере. GitHub позволяет просматривать и скачивать материалы. Установка и первый запуск описаны в `START_HERE.md`.

Программная среда

Для работ № 1 и № 2 нужны Python, NumPy, pandas, scikit-learn и Matplotlib; для второго примера также TensorFlow/Keras. Для работы № 3 нужны PyTorch, совместимая с ним torchvision и Pillow. Инструкции по проверенному локальному окружению находятся в `code/README.md`.

CPU (англ. central processing unit) – центральный процессор компьютера. **GPU (англ. graphics processing unit)** – графический процессор; он ускоряет многие операции нейросетей, особенно обработку групп изображений. Для его использования нужны совместимые версии Python, PyTorch, torchvision и программного обеспечения ускорителя. Сначала проверьте версии установленных библиотек:

```
1 import sys
2 from importlib.metadata import version
3 print(sys.version)
4 for name in ("numpy", "pandas", "scikit-learn"):
5     print(name, version(name))
```

CUDA – программная платформа для вычислений на GPU NVIDIA. Проверка `torch.cuda.is_available()` сообщает о ее доступности для PyTorch. Объем памяти нужно оценить отдельно: в третьей работе расчет параметров исходной сети позволяет сделать это до длительного обучения. Небольшую проверку кода можно выполнить на CPU.

Пути в ноутбуке

Первая ячейка находит корень проекта по папкам `code` и `labs`, затем выбирает каталог работы. Короткие пути `data/train.csv` и `runs` отсчитываются от него. Поэтому ноутбук можно открыть из корня проекта или из каталога работы, сохранив структуру папок.

> **Перед запуском**
>
> Выполните ячейку подготовки и проверьте выведенный рабочий каталог. Затем прочитайте таблицу или изображение и убедитесь, что получили данные нужной работы.

Практическая работа № 1. Классификация

1.1. Задача распознавания активности

По признакам сигналов смартфона требуется определить вид активности человека. Каждый объект – временное окно измерений, то есть короткий фрагмент записи датчиков. Оно представлено вектором $x_i \in \mathbb{R}^d$ из d числовых характеристик. Его истинная метка y_i известна из записи эксперимента; модель с параметрами θ выдает прогноз $\widehat{y}_i$:

$$\widehat{y}_i = f_\theta(x_i), \qquad y_i \in \{0, 1, 2, 3, 4, 5\}. \tag{1.1}$$

Коды классов обозначают шесть категорий активности. Сходство двух движений оценивают по признакам сигналов; расстояние между числовыми кодами меток физического смысла не имеет.

Таблица 2: Словарь меток активности

Код	Метка в файле	Смысл
0	`WALKING`	Ходьба
1	`WALKING_UPSTAIRS`	Ходьба вверх по лестнице
2	`WALKING_DOWNSTAIRS`	Ходьба вниз по лестнице
3	`SITTING`	Сидение
4	`STANDING`	Стояние
5	`LAYING`	Положение лежа

1.2. От сигналов к признакам

В задании используется набор распознавания активности UCI HAR (англ. Human Activity Recognition Using Smartphones). В первоисточнике 30 участников выполняли шесть действий со смартфоном на поясе; акселерометр и гироскоп регистрировали сигналы с частотой 50 Гц. Окна длиной 2,56 с содержали 128 отсчетов и перекрывались на 50%. Из каждого окна извлекался 561 признак во временной и частотной областях [5].

Акселерометр измеряет ускорение, гироскоп – угловую скорость. При подготовке сигналов фильтрация ослабляет шум, а выделение медленно меняющейся, низкочастотной составляющей помогает разделить вклад гравитации и движения тела. Производные характеризуют скорость изменения сигнала; евклидова норма – длина вектора – объединяет три оси в одно число. Преобразование Фурье переводит сигнал из временной области, где описано изменение измерений во времени, в частотную: она показывает вклад колебаний разных частот. **Быстрое преобразование Фурье** (БПФ; англ. fast Fourier transform, FFT) – алгоритм вычисления такого представления. Учебные файлы содержат готовые признаки этой обработки.

Медиана (англ. median) – центральное значение упорядоченного набора; при четном числе значений берется среднее двух центральных.

Таблица 3: Как читать обозначения признаков исходного задания

Обозначение	Что оно помогает описать
`tBodyAcc`, `tGravityAcc`, `tBodyGyro`	Ускорение тела, гравитационная составляющая, угловая скорость; `t` обозначает временную область.
`Jerk`, `Mag`, `XYZ`	Изменение сигнала, его модуль, отдельные оси.
Префикс `f`	Представление в частотной области после БПФ.
`mean`, `std`, `mad`, `iqr`	Среднее, стандартное отклонение, медианное абсолютное отклонение и межквартильный размах.
`max`, `min`, `sma`, `energy`, `entropy`	Экстремумы, совокупная амплитуда, энергия и характеристика распределения сигнала.
`arCoeff`, `correlation`, `angle`	Коэффициенты связи сигнала с его прошлыми значениями, корреляция сигналов, угол между векторами.
`maxInds`, `meanFreq`, `bandsEnergy`, `skewness`, `kurtosis`	Положение частотных компонентов, энергия полос и форма распределения.

Название `energy` указывает на квадраты значений сигнала, но точная величина зависит от нормирования. Для последовательности $a_1, \ldots, a_N$ нормированная энергия может определяться как $N^{-1} \sum_i a_i^2$, где N – число отсчетов. Например, для значений $1, -1, 2, -2$ сумма квадратов равна $1 + 1 + 4 + 4 = 10$, а после деления на число отсчетов получается $10/4 = 2{,}5$. Поэтому интерпретация столбца требует его определения в описании признаков.

1.3. Разделение данных по участникам

Соседние окна исходной записи имеют 64 общих отсчета из 128. Если такие окна одного человека попадают и в обучение, и в проверку, модель встречает близкие фрагменты записи и индивидуальные особенности уже знакомого участника. При применении к новому человеку эти условия меняются. Поэтому единица разделения в данной работе – **человек**, а единица классификации – **окно сигнала**: все окна участника относятся к одной части данных.

`GroupShuffleSplit` выбирает долю групп. Участники могут иметь разное число окон, поэтому доля строк в валидации не обязана совпадать с долей участников.

Масштабирование признаков (англ. feature scaling) меняет их числовые масштабы, чтобы единицы измерения не определяли результат сравнения. В этой работе используется **стандартизация** (англ. standardization): $\widetilde{x}_j = (x_j - \mu_j)/s_j$. Здесь μ_j – среднее обучающего признака, а s_j – его стандартное отклонение для непостоянного столбца. Например, при $\mu_j = 10$ и $s_j = 2$ проверочное значение

$x_j = 14$ преобразуется в $(14 - 10)/2 = 2$. Объект `StandardScaler` сохраняет эти статистики и применяет преобразование к новым данным; далее будем называть его стандартизатором.

Обучающие и проверочные объекты используют одни и те же μ_j и s_j. Их пересчет на проверочной части использовал бы ее распределение при подготовке модели. **Конвейер обработки** (англ. pipeline) связывает последовательные операции подготовки данных и прогнозирования. Объект `Pipeline` хранит стандартизатор вместе с классификатором: при `fit` на обучающей части находит статистики преобразования, при прогнозе применяет сохраненные значения.

1.4. Три семейства классификаторов

Метод ближайших соседей

Метод k ближайших соседей (англ. k-nearest neighbors, KNN) находит для нового вектора k ближайших обучающих объектов и определяет класс голосованием. При евклидовом расстоянии

$$\rho(x, z) = \sqrt{\sum_{j=1}^{d} (x_j - z_j)^2} \tag{1.2}$$

вклад каждой координаты входит в сумму квадратов. Например, разности 100 и 1 по двум признакам дают $\rho^2 = 10000 + 1$: первый признак почти полностью определяет расстояние. Изменение единицы его измерения может изменить состав ближайших соседей. Поэтому KNN в исходном сравнении обучается после стандартизации.

При малом k отдельный ошибочно размеченный или необычный объект сильнее влияет на голосование. Увеличение k сглаживает границу, но может включить в окрестность объекты другого класса. При равных весах каждый сосед дает один голос; взвешивание по расстоянию усиливает влияние ближайших. Метрика расстояния задает само понятие близости; ее следует отличать от метрики качества прогноза.

Линейный метод опорных векторов

Метод опорных векторов (англ. support vector machine, SVM) строит границу разделения классов. Для двух классов линейная граница задается условием $w^\mathsf{T} x + b = 0$, где w – вектор коэффициентов, а b – свободный член. При двух признаках это прямая: знак выражения определяет сторону, на которой лежит объект. В большей размерности сохраняется то же правило. SVM ищет границу с широким зазором между классами, допуская штрафуемые нарушения разделения.

Параметр C задает вес этого штрафа относительно **регуляризации** (англ. regularization), которая ограничивает сложность решения, здесь – величину коэффициентов w. При уменьшении C регуляризация усиливается, а нарушения разделения обходятся модели дешевле. Последствием могут стать

как уменьшение переобучения, так и утрата полезных различий между классами; направление изменения качества устанавливается на валидации. Для шести активностей `LinearSVC` обучает линейные разделители по схеме «один класс против остальных»: каждый разделитель отличает один вид активности от всех других.

Случайный лес

Дерево решений (англ. decision tree) последовательно сравнивает признаки с порогами и выдает класс в конечном узле. Более глубокое дерево может описать сложную границу, но также выделить малые группы, характерные только для обучения.

Случайный лес (англ. random forest) – ансамбль (англ. ensemble), то есть группа моделей, чьи прогнозы объединяются. Он усредняет оценки классов, полученные деревьями. При обучении деревьев случайно выбираются объекты, а при разбиениях – доступные признаки. Различия между деревьями помогают уменьшить влияние ошибок отдельного дерева.

Глубина ограничивает сложность каждого дерева, а число деревьев влияет на устойчивость результата и вычислительные затраты. Пороговое условие сохраняет смысл при стандартизации: например, $x_j > 14$ при $\mu_j = 10$, $s_j = 2$ соответствует $\widetilde{x}_j > 2$. Поэтому лес в исходном сравнении используется без стандартизатора.

Значения k, C, глубины и числа деревьев фиксируются до обучения; коэффициенты линейной модели и пороги дерева определяются при `fit`. Смена гиперпараметров требует нового обучения. Базовой моделью служит `DummyClassifier`, всегда предсказывающий самый частый класс обучения. Его результат показывает, какого качества можно достичь, используя только частоты меток. Сравнение с этой моделью помогает оценить пользу признаков.

1.5. Метрики и матрица ошибок

Матрица ошибок (англ. confusion matrix) показывает, какие классы модель распознает верно и какие путает. Пусть C_{ij} – число объектов истинного класса i, отнесенных к классу j. Строки соответствуют истине, столбцы – прогнозу. Для класса c:

$$TP_c = C_{cc}, \qquad FP_c = \sum_{i \neq c} C_{ic}, \qquad FN_c = \sum_{j \neq c} C_{cj}, \tag{1.3}$$

$$P_c = \frac{TP_c}{TP_c + FP_c}, \qquad R_c = \frac{TP_c}{TP_c + FN_c}, \qquad F_{1,c} = \frac{2TP_c}{2TP_c + FP_c + FN_c}. \tag{1.4}$$

При оценке класса c положительным прогнозом считается выбор этого класса. Здесь TP_c – верные прогнозы класса (англ. true positives), FP_c – объекты других классов, ошибочно отнесенные к нему (англ. false positives), FN_c – объекты этого класса, которые модель пропустила (англ. false negatives).

Точность положительных прогнозов (англ. precision) P_c – доля верных ответов среди всех прогнозов класса c. **Полнота** (англ. recall) R_c – доля найденных объектов среди всех истинных объектов этого класса. В первом знаменателе учитываются ложные положительные ответы FP_c, во втором – пропущенные объекты FN_c. **F1-мера** (англ. F1 score) объединяет точность и полноту и уменьшается, если хотя бы одна из них мала.

Для общей оценки шести классов используем **макроусреднение** (англ. macro averaging) – среднее значений $F_{1,c}$, обозначаемое F_1 (macro-F1).

$$\text{Accuracy} = \frac{\sum_c C_{cc}}{n}, \qquad F_1 = \frac{1}{6}\sum_{c=0}^{5} F_{1,c}. \tag{1.5}$$

Доля верных ответов (англ. accuracy) вычисляется по всей выборке сразу; ее не следует смешивать с точностью для отдельного класса. При макроусреднении каждый класс имеет одинаковый вес, поэтому малочисленный класс влияет на итоговую F1 так же, как многочисленный. **Взвешенная F1** (англ. weighted-F1) использует веса, пропорциональные числу объектов классов. В отчете это число указано в столбце `support`. Если модель хорошо распознает частые активности и пропускает редкую, accuracy и weighted-F1 могут скрыть размер проблемы. Поэтому основная метрика выбора в данной работе – macro-F1 валидации, а accuracy и показатели по классам дополняют анализ.

> **Расчет метрик одного класса**
>
> Пусть для класса «сидит» модель дала 10 положительных прогнозов: восемь верны, два относятся к другим классам; еще четыре объекта класса «сидит» пропущены. Получаем $TP = 8$, $FP = 2$, $FN = 4$. Тогда precision равна $8/10 = 0{,}8$, recall – $8/12 \approx 0{,}667$, F1 – $16/22 \approx 0{,}727$.

При нулевом знаменателе метрика не определена обычной формулой. В коде `zero_division=0` задает явное соглашение вывода; нулевое число объектов класса или отсутствие предсказаний следует отметить. Нормирование матрицы по строкам означает деление каждой ячейки на число объектов соответствующего истинного класса. Оно помогает увидеть долю ошибок для каждого класса, а абсолютные числа показывают объем наблюдений.

1.6. Порядок выполнения

Шаг 1. Проверьте таблицы и метки

Откройте файлы из `data`. Выведите размеры таблиц, имена столбцов, число участников, распределение классов и наличие пропусков. Найдите `Activity` и `subject`. Признаки выбирайте по именам: срез `values[:, :-2]` зависит от текущего порядка служебных столбцов. Для двух признаков приведите определение из описания данных и объясните, какое свойство сигнала они характеризуют.

```python
from pathlib import Path
import pandas as pd
train = pd.read_csv(Path("data/train.csv"))
test = pd.read_csv(Path("data/test.csv"))
print(train.shape, test.shape)
print(train["Activity"].value_counts())
print(train["subject"].nunique(), test["subject"].nunique())
```

Шаг 2. Выделите валидацию

Отведите внешний `test.csv` для единственной итоговой оценки выбранной модели. Из участников `train.csv` выделите примерно 20% в валидацию с фиксированным seed 40. Выбор seed по наилучшей метрике добавил бы подбор самого разбиения.

Учебный модуль проверяет метки и признаки, исключает целевой столбец и идентификатор участника, затем выполняет групповое разделение. Если идентификатор называется иначе, укажите его через `group_column`. При его отсутствии уточните состав выгрузки и протокол: фиктивные номера, полученные из номеров строк, не позволяют проверить перенос на новых людей.

```python
from lab1_utils import load_har, candidate_models,
    fit_validation
from lab1_utils import CLASSES, final_test, save_har_result
data = load_har("data", group_column="subject", seed=40)
print(data["Xtr"].shape, data["Xva"].shape)
print(data["training_subjects"], data["validation_subjects"])
```

> **Проверка разбиения и входов**
>
> Проверьте, что каждая часть содержит все шесть классов, участники обучения и валидации не пересекаются, а участники внешнего теста отсутствуют в обеих частях. Во входах нет `Activity` и идентификатора участника. Столбцы теста приведены к порядку обучающих признаков по именам. Стандартизатор обучается только на обучающей части; валидация и тест используют сохраненное преобразование.

Шаг 3. Сравните конфигурации

До запуска сформулируйте ожидаемое влияние k, C и глубины деревьев. Выполните исходный план из таблицы, затем переходите к собственным опытам. Допускается иной набор нескольких алгоритмов с сопоставимым и заранее определенным объемом исследования. Используйте библиотечные реализации; самостоятельная реализация алгоритмов не требуется.

Таблица 4: Исходный план сравнения классификаторов

Обозначения	Модель	Меняемый параметр
D0	DummyClassifier	Самый частый класс; ориентир без использования признаков
K3, K7, K15	KNN	$k \in \{3, 7, 15\}$; одинаковое масштабирование
S0.1, S1, S10	LinearSVC	$C \in \{0{,}1, 1, 10\}$; остальные настройки фиксированы
R8, RNone	Случайный лес	Глубина 8 или без ограничения; 200 деревьев

```
models = candidate_models(seed=40)
rows = []
for name, estimator in models.items():
    result = fit_validation(data, estimator)
    rows.append({"model": name, **result["metrics"],
                 "seconds_fit": result["seconds_fit"]})
comparison = pd.DataFrame(rows)
print(comparison.sort_values("macro_f1", ascending=False))
```

Для всех конфигураций цикл использует одно разбиение на обучение и валидацию. Выберите строку таблицы и объясните, какие прогнозы входят в ее macro-F1. Сохраняйте предупреждения о сходимости (англ. convergence): они сообщают, что алгоритм настройки не достиг установленного условия остановки за доступное число шагов. В этом случае выясните причину и отдельно зафиксируйте изменение бюджета или настройки; общее отключение `warnings` скрыло бы различие условий сравнения.

Шаг 4. Визуализируйте и зафиксируйте выбор

Постройте столбчатую диаграмму macro-F1 всех конфигураций и отдельную диаграмму времени обучения: эти величины имеют разные единицы и смысл. По валидационным показателям назовите хуже всего распознаваемые классы и возможные причины ошибок. Выберите конфигурацию по macro-F1 валидации. При близких результатах учитывайте сложность и время; вывод об устойчивости небольшого различия требует повторной проверки в заранее заданном протоколе.

```
import matplotlib.pyplot as plt
comparison.plot.bar(x="model", y="macro_f1", legend=False)
plt.ylabel("Validation macro-F1")
plt.tight_layout()
```

Шаг 5. Выполните итоговый тест

Зафиксируйте выбранную конфигурацию, затем переобучите ее на всем `train.csv`. Стандартизатор внутри `Pipeline` также обучается заново: после

выбора модели бывшая валидация становится частью обучающих данных. Получите один итоговый прогноз `yhat` для `test.csv` и сохраните результат. Учебный модуль формирует отчет по классам с помощью `classification_report`: он содержит метрики каждого класса и число его объектов. Код ниже выбирает максимум macro-F1 валидации; при равенстве метрик порядок определяется именем модели.

```
selected = comparison.sort_values(
    ["macro_f1", "model"], ascending=[False, True]
).iloc[0]["model"]
final = final_test(data, models[selected])
yhat = final["prediction"]
print(pd.DataFrame(final["report"]).T)
save_har_result(data, comparison, selected, final, "runs/
   lab1_01")
```

Разберите полученное тестовое качество, даже если оно хуже ожидаемого. Возвращение к подбору настроек после просмотра теста использует тестовые ответы для выбора модели и лишает оценку независимости.

Шаг 6. Разберите ошибки

Постройте матрицу ошибок для итоговых прогнозов с одинаковым порядком шести меток. Для ее изображения используют **тепловую карту** (англ. heatmap): цвет ячейки соответствует числу или доле объектов. Выведите абсолютные числа и нормирование по строкам. При чтении элемента C_{ij} назовите истинное действие i и ошибочный прогноз j. Разберите хотя бы две заметные пары перепутанных действий; отдельно укажите наблюдаемую ошибку и предполагаемую причину.

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(
    data["ytest"], yhat, labels=range(6),
    display_labels=CLASSES, normalize="true", xticks_rotation
   =90
)
plt.tight_layout()
```

1.7. Отчет и вопросы защиты

В отчете по работе № 1 должны быть паспорт файлов и объяснение двух признаков, состав разбиения по участникам, мотивация выбора алгоритмов, таблица всех конфигураций, диаграммы macro-F1 и времени обучения, итоговый `classification_report` и обе формы матрицы ошибок. Объясните precision, recall и F1 на примере одного класса. Сохраните точные параметры модели, версии библиотек, seed и идентификаторы групп. В выводе объясните влияние

гиперпараметров, укажите выбранный вариант, хуже всего распознаваемые классы и следующий проверяемый вопрос. Вывод о новых людях ограничьте проверенными группами и условиями записи.

1. Что является объектом, признаком, целевой меткой и группой в этой работе?
2. Почему код активности нельзя трактовать как непрерывную величину?
3. Какую информацию дают акселерометр и гироскоп? Зачем нужны признаки частотной области?
4. Почему разбиение строк и разбиение участников проверяют разные условия применения?
5. Чем precision отличается от recall? Постройте собственный пример с FP и FN.
6. Как рассчитывается F1? Чем macro-F1 отличается от weighted-F1 и accuracy?
7. Чем гиперпараметр отличается от параметра обученной модели?
8. Как изменение k, C и глубины деревьев повлияло на ваши результаты?
9. Почему стандартизатор обучается только на обучающей части? Для каких моделей он особенно существенен?
10. Что означает нулевой support? Почему следует сохранять порядок меток в отчете?
11. Почему нельзя выбрать победителя по внешнему тесту и затем назвать этот тест независимым?

Практическая работа № 2. Нейронные сети

2.1. Постановка задачи

По характеристикам дома требуется предсказать `SalePrice`. Для дома i подготовленные признаки образуют вектор x_i, известная цена обозначается y_i, а сеть f_θ с параметрами θ вычисляет прогноз:

$$\widehat{y}_i = f_\theta(x_i), \qquad y_i \in \mathbb{R}_{>0}. \tag{2.1}$$

В этой работе мы подготовим табличные данные, обучим полносвязную сеть и исследуем, как ее настройки влияют на ошибку. Выбранная конфигурация даст прогноз для домов из `test.csv`.

2.2. Как используются исходные таблицы

В `train.csv` цены известны. Его строки делятся на обучающую часть (80%) и валидационную (20%): по первой настраиваются преобразования и веса сети, по второй сравниваются конфигурации. Файл `test.csv` содержит дома без известных цен, поэтому служит для получения прогнозов и не позволяет измерить их ошибку.

Подготовка данных тоже использует информацию из примеров. Например, медиана зависит от набора строк, по которому ее вычислили. Если найти ее по всему `train.csv` до разделения, в подготовку обучения попадут сведения о валидации. Если же независимо кодировать категории в разных таблицах, один числовой код может получить разные значения. Для корректного сравнения нужна одна система преобразований.

> **Предобработка и применение модели**
>
> Медианы, средние, масштабы и словарь категорий определяются по обучающей части. Валидация и новые дома преобразуются с уже найденными параметрами: `fit_transform` на обучении, `transform` на остальных данных.

Случайное разделение здесь опирается на учебное предположение о независимых однородных наблюдениях. Временной прогноз, повторные продажи одного объекта или группы зависимых домов потребовали бы другого разбиения. Это предположение необходимо указать в отчете.

2.3. Пропуски, категории и масштабирование

Числовые пропуски в исходной конфигурации заменяются медианой обучающей части. По сравнению со средним медиана меньше реагирует на отдельные очень большие значения, поэтому такие наблюдения слабее влияют на заполнение пропусков.

Категориальные пропуски получают значение `__missing__`. При этом описание данных помогает различить неизвестное значение и отсутствие объекта, например гаража. Эти случаи могут иметь разный смысл для цены. В

исходной конфигурации по условию задания удаляются пять столбцов: `Alley`, `FireplaceQu`, `PoolQC`, `Fence`, `MiscFeature`. Фиксированный список обеспечивает одинаковые условия опытов; другие способы работы с этими признаками можно исследовать отдельно.

Категории без естественного порядка кодируются методом **бинарных индикаторов (англ. one-hot encoding)**: каждой известной категории соответствует отдельный бинарный столбец. Например, число, обозначающее вид покрытия, не должно само по себе задавать расстояние между видами.

> **Кодирование трех категорий**
>
> Для категорий A, B, C возможны коды $A \mapsto (1,0,0)$, $B \mapsto (0,1,0)$, $C \mapsto (0,0,1)$. Замена их числами $1, 2, 3$ навязывает порядок и расстояния, которых в описании признака может не быть.

Обученный кодировщик задает состав и порядок столбцов. Для неизвестных категорий используется `handle_unknown="ignore"`: они получают нулевой блок индикаторов. Это позволяет вычислить прогноз, но модель не приобретает знаний о новой категории. Ее появление следует отметить при анализе данных.

Числовые признаки приводятся к сопоставимому масштабу:

$$\widetilde{x}_{ij} = \frac{x_{ij} - \mu_j}{s_j}, \tag{2.2}$$

где μ_j, s_j вычислены по обучающей части после заполнения пропусков. Для постоянного признака реализация использует масштаб 1, чтобы избежать деления на ноль. Стандартизация меняет масштаб, но сохраняет зависимости между признаками и форму их распределений; она не делает распределение нормальным.

Целевая цена в примере также стандартизуется:

$$z_i = \frac{y_i - \mu_y}{s_y}, \qquad \widehat{y}_i = \mu_y + s_y \widehat{z}_i. \tag{2.3}$$

Сеть обучается предсказывать z_i, то есть работает с числами умеренного масштаба. Для оценки ошибки в денежных единицах и сохранения прогнозов применяется обратное преобразование с теми же μ_y, s_y, которые найдены на обучении. Средняя цена валидации в это преобразование не входит.

В статье Uber Engineering от 5 сентября 2017 года описано, как в Michelangelo правила преобразования признаков входят в конфигурацию модели и одинаково применяются при обучении и прогнозировании [6]. В лабораторной тот же принцип согласованности требует сохранять обученную предобработку вместе с моделью: новые дома должны проходить через прежние преобразования.

2.4. Полносвязная нейронная сеть

Нейронная сеть (англ. neural network) объединяет настраиваемые вычислительные элементы – нейроны – в слои. В **полносвязном слое (англ. fully connected layer)** каждый нейрон получает все выходы предыдущего слоя. Сеть с одним или несколькими скрытыми слоями называют **многослойным перцептроном (англ. multilayer perceptron, MLP)**. Скрытые слои находятся между входом и выходом; их ответы служат промежуточными признаками для следующих слоев.

Нейрон полносвязного слоя вычисляет взвешенную сумму входов и добавляет смещение. **Функция активации (англ. activation function)** преобразует эту сумму перед передачей следующему слою:

$$a_j = \sum_{k=1}^{d} w_{jk} x_k + b_j, \qquad h_j = \varphi(a_j). \tag{2.4}$$

В матричной записи для слоя ℓ:

$$h^{(\ell)} = \varphi\left(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}\right), \qquad h^{(0)} = \widetilde{x}. \tag{2.5}$$

Выходы одного слоя становятся входами следующего. Для сети с одним скрытым слоем и линейным выходом получаем

$$\widehat{z} = v^{\mathsf{T}} \varphi(W\widetilde{x} + b) + c. \tag{2.6}$$

В Keras такую последовательность задает `keras.Sequential`, а полносвязный слой представлен `Dense`. До обучения весам задаются начальные значения – это **инициализация (англ. initialization)**.

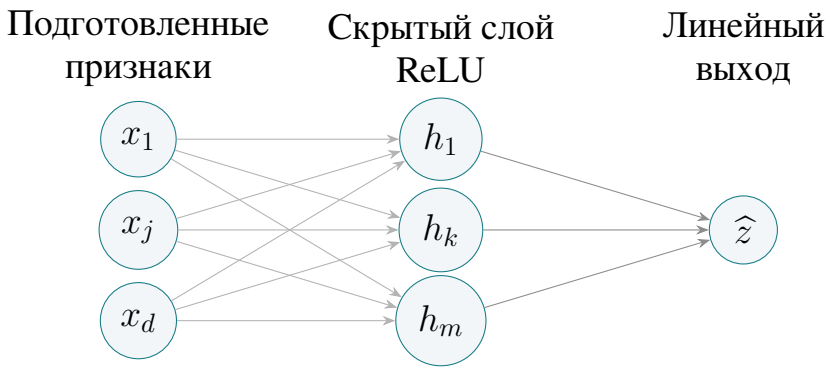


Рис. 2: Схема полносвязной сети. Число входов определяется после предобработки; на схеме показана часть связей.

Входная размерность d определяется числом подготовленных признаков. Изменение ширины **первого скрытого полносвязного слоя** меняет число нейронов, обрабатывающих этот вектор, и сохраняет число его компонент. На выходе нужен один нейрон, поскольку для каждого дома предсказывается одна цена.

У слоя с d входами и m нейронами имеется $dm+m$ обучаемых параметров: dm весов и m смещений. Для схемы $d \to m \to 1$ их всего

$$P = (d+1)m + (m+1). \tag{2.7}$$

Например, при $d = 10$, $m = 8$ получаем $P = 97$: первый слой содержит 88 параметров, выходной – 9. В файлах курса размерность определяется предобработкой; для нее расчет можно проверить командой `model.count_params()`.

2.5. Функции активации

Без нелинейной активации последовательность линейных слоев сводится к одному аффинному преобразованию – взвешенной сумме входов со смещением. Нелинейность позволяет сети описывать более сложные зависимости.

Активация	Формула	Особенность
ReLU	$\max(0, a)$	Выпрямляющая линейная функция (англ. rectified linear unit); на отрицательной полуоси производная равна нулю.
Сигмоида (англ. sigmoid)	$1/(1+e^{-a})$	Выход в $(0, 1)$; в областях насыщения производная мала.
tanh	$\tanh(a)$	Гиперболический тангенс; выход в $(-1, 1)$, также имеет области насыщения.
Линейная	a	Не ограничивает диапазон прогноза.

В исходной сети используется ReLU в скрытом слое и линейная функция на выходе. Сигмоида без согласованного преобразования цели ограничила бы прогноз интервалом $(0, 1)$. Стандартизованная цена может быть отрицательной: такое значение соответствует цене ниже μ_y. Допустимость прогнозируемой цены проверяется после обратного преобразования.

PReLU и другие модификации ReLU

Параметрическая ReLU (англ. parametric rectified linear unit, PReLU) сохраняет линейную ветвь для положительных входов, а для отрицательных использует наклон α:

$$\mathrm{PReLU}(a) = \begin{cases} a, & a \geq 0, \\ \alpha a, & a < 0. \end{cases} \tag{2.8}$$

Коэффициент α обучается вместе с весами сети. Например, при $a = -2$ и $\alpha = 0{,}1$ выход равен $-0{,}2$, тогда как ReLU выдает ноль. На отрицательной ветви производная PReLU равна α, а у ReLU – нулю. У близкой модификации LeakyReLU (англ. leaky rectified linear unit) наклон задается заранее и не обучается. В PyTorch PReLU задает модуль `nn.PReLU`.

Экспоненциальная линейная функция (англ. exponential linear unit, ELU) имеет другую отрицательную ветвь:

$$\mathrm{ELU}(a) = \begin{cases} a, & a \geq 0, \\ \alpha(e^a - 1), & a < 0, \end{cases} \qquad \alpha > 0. \tag{2.9}$$

При больших отрицательных входах ее выход приближается к $-\alpha$. Здесь α – задаваемый гиперпараметр. Выбор между ними проверяют по валидационной ошибке при одинаковых условиях обучения.

Softmax и вероятности классов

В классификации последний слой выдает по одному числу для каждого из K классов. Эти числа $z_1, \ldots, z_K$ называют **логитами** (англ. logits); они могут быть любыми вещественными числами. **Нормированная экспонента** (англ. softmax) преобразует их в предсказанные вероятности классов:

$$p_c = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}, \qquad c = 1, \ldots, K, \qquad \sum_{c=1}^{K} p_c = 1. \tag{2.10}$$

В отличие от ReLU, softmax обрабатывает весь вектор оценок одного объекта: знаменатель связывает выходы для разных классов. Например, логиты $(2, 1, 0)$ дают вероятности примерно $(0{,}665;\ 0{,}245;\ 0{,}090)$. Модель выбирает первый класс, поскольку у него наибольшая вероятность. Softmax сохраняет порядок логитов, поэтому для выбора класса достаточно максимальной исходной оценки. В PyTorch для матрицы `(batch, classes)` используется `nn.Softmax(dim=1)`.

Softmax применяется к выходам классификатора. Для сети с одним выходом он всегда дает 1 и потому не подходит для прогнозирования цены. В работе № 2 выход остается линейным, а использование вероятностей классов будет рассмотрено в работе № 3.

2.6. Функции потерь и метрики регрессии

Функция потерь (англ. loss function, далее loss) измеряет ошибку, которую алгоритм уменьшает при обучении. В регрессии используют среднюю квадратичную ошибку (англ. mean squared error, MSE), уже знакомую MAE и корень из средней квадратичной ошибки (англ. root mean squared error, RMSE). MSE и MAE могут служить как функциями потерь, так и метриками; роль задается тем, используется ли величина для изменения весов или для оценки

результата. Для n оцениваемых объектов обозначим ошибку $e_i = y_i - \widehat{y}_i$:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} e_i^2, \tag{2.11}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |e_i|, \tag{2.12}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}. \tag{2.13}$$

MAE и RMSE измеряются в единицах цены, MSE – в квадрате этих единиц. Квадрат ошибки усиливает вклад крупных промахов, тогда как в MAE вклад растет пропорционально абсолютной величине ошибки.

Пример расчета метрик регрессии

Пусть цены равны $(100, 150, 200)$, а прогнозы – $(110, 140, 230)$ в условных денежных единицах. Абсолютные ошибки составляют 10, 10 и 30, а их квадраты – 100, 100 и 900. Поэтому $\mathrm{MAE} = 50/3 \approx 16{,}67$, $\mathrm{MSE} = 1100/3 \approx 366{,}67$, $\mathrm{RMSE} \approx 19{,}15$.

Дополнительно можно вычислить коэффициент детерминации

$$R^2 = 1 - \frac{\sum_i (y_i - \widehat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}. \tag{2.14}$$

Здесь $\overline{y}$ – среднее по *оцениваемой* части. Оно входит в определение метрики и не используется для предобработки. Значение R^2 может быть отрицательным и не выражает процент точных ответов. При постоянной цели знаменатель равен нулю, поэтому формула не определена.

Обучение минимизирует выбранную функцию потерь; конфигурации сравниваются по общим метрикам. Поэтому при смене MSE на MAE значения `loss` нельзя сопоставлять напрямую. Во всех опытах главным критерием служит MAE валидации в исходных единицах цены, а RMSE помогает оценить влияние крупных ошибок.

Для преобразования (2.3) справедливо $\mathrm{MAE}_y = s_y\,\mathrm{MAE}_z$ и $\mathrm{MSE}_y = s_y^2\,\mathrm{MSE}_z$. Из-за этого MAE в журнале обучения на стандартизованной цели отличается по численному значению от MAE в денежных единицах.

2.7. Обучение: градиент, эпоха и batch

Чтобы уменьшить значение функции потерь, алгоритм изменяет веса и смещения сети. **Градиент (англ. gradient)** – вектор производных функции потерь по параметрам. Он показывает направление ее наиболее быстрого роста при малом изменении параметров. В обычном градиентном методе обновление направлено в противоположную сторону, а его величину регулирует скорость

обучения. За один шаг обычно используют часть обучающих объектов – **batch (англ. mini-batch; мини-выборку)**. Для batch B_t шаг имеет вид

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \frac{1}{|B_t|} \sum_{i \in B_t} \ell(y_i, f_\theta(x_i)), \tag{2.15}$$

где $\eta > 0$ – **скорость обучения (англ. learning rate)**. Для вычисления производных используется цепное правило. Передачу производных от выхода сети к предшествующим слоям называют **обратным распространением ошибки (англ. backpropagation)**.

Один шаг обучения

Пусть $\widehat{y} = wx + b$, $x = 2$, правильный ответ $y = 6$, начальные $w = 1$, $b = 0$. Прогноз равен 2, а значение функции потерь $(\widehat{y} - y)^2 = 16$. Производные:

$$\frac{\partial L}{\partial w} = 2(\widehat{y} - y)x = -16, \qquad \frac{\partial L}{\partial b} = 2(\widehat{y} - y) = -8.$$

При шаге $0{,}01$ получаем $w = 1{,}16$, $b = 0{,}08$. Новый прогноз равен $2{,}4$, loss – $12{,}96$: обновление уменьшило ошибку этого объекта. В многослойной сети тот же принцип требует вычислить производные через все слои.

Итерация (англ. iteration) здесь соответствует одному обновлению весов по batch, **эпоха (англ. epoch)** – проходу по всей обучающей части. Если в ней N строк и последний неполный batch сохраняется, число обновлений за эпоху равно $\lceil N/b \rceil$, где b – `batch_size`. При $N = 1000$, $b = 32$ это 32 обновления, а за 10 эпох – 320.

Меньший batch дает больше обновлений за эпоху и обычно более шумную оценку градиента. Поэтому равное число эпох при разных batch задает разное число обновлений. Скорость обучения тоже влияет на ход процесса: слишком большой шаг может вызвать колебания или расходимость, слишком малый – замедлить улучшение.

Оптимизатор (англ. optimizer) задает правило обновления параметров. В работе сравниваются стохастический градиентный спуск (англ. stochastic gradient descent, SGD) и Adam (англ. adaptive moment estimation). SGD использует градиент текущего batch; Adam накапливает оценки первого и второго моментов градиента и адаптирует масштаб обновлений. Одинаковая численная скорость обучения дает этим алгоритмам разные траектории и может подходить им в разной степени.

2.8. Переобучение и воспроизводимость

Если ошибка обучения продолжает уменьшаться, а ошибка валидации перестает улучшаться или растет, возникает подозрение на переобучение. Кривые показывают расхождение поведения на двух частях данных, но для выяснения

причины нужно также проверить подготовку данных, размер выборки, архитектуру и скорость обучения.

Штраф L2 добавляет к функции потерь слагаемое, растущее с квадратами весов. **Случайное отключение (англ. dropout)** при обучении исключает часть выходов скрытого слоя; при прогнозировании отключение прекращается. Влияние обоих приемов проверяется на валидации; исходная конфигурация их не содержит.

Ранний останов (англ. early stopping) прекращает обучение после заданного числа эпох без улучшения валидационной метрики. В Keras его реализует `EarlyStopping`; возврат лучших сохраненных весов задается параметром `restore_best_weights=True`. В основных опытах ранний останов выключен: сравниваются результаты после заданного числа проходов. Его влияние исследуется отдельно.

В протоколе используется одно разбиение и исходный seed 40. Команда `keras.utils.set_random_seed` задает начальные состояния генераторов; включение детерминированных операций дополнительно уменьшает случайность вычислений. Для повторения опыта нужны также версии библиотек и сведения об устройстве. Один seed не оценивает устойчивость результата и не гарантирует побитового совпадения на разных программных и аппаратных платформах.

2.9. Порядок выполнения

Сначала выполните B0: проследите путь одного дома от строки таблицы через подготовку признаков до прогноза и объясните полученную MAE в денежных единицах. После этого переходите к сравнению конфигураций.

Шаг 1. Изучите исходные таблицы

Откройте таблицы и описание данных. Проверьте число строк и столбцов, уникальность `Id` внутри каждого файла и наличие `SalePrice` только в `train.csv`. Выберите два числовых и два категориальных признака и объясните их смысл.

```
from pathlib import Path
import pandas as pd

DATA_DIR = Path("data")
train_data = pd.read_csv(DATA_DIR / "train.csv")
test_data = pd.read_csv(DATA_DIR / "test.csv")
print(train_data.shape, test_data.shape)
print(train_data.head())
print(test_data.head())
print(train_data.isna().sum().sort_values(ascending=False).
    head(10))
```

`Id` связывает прогноз с домом и исключается из признаков. Если в размеченной строке отсутствует целевая цена, ее нельзя заменять медианой: для этой строки нет достоверного ответа, по которому можно обучать модель.

Шаг 2. Разделите и подготовьте данные

Отделите `SalePrice` и `Id`, сохраните удаление пяти заданных столбцов и разделите сырые строки в отношении 80/20 с `random_state=40`. Затем обучите предобработку на обучающей части. Эти действия реализованы в `lab2_utils.py`; в функции `prepare_frames` найдите, где вычисляются статистики.

```
import sys
sys.path.insert(0, "code")
from lab2_utils import load_course_data, constant_baselines

data = load_course_data(DATA_DIR, seed=40)
print(data["Atr"].shape, data["Ava"].shape, data["Atest"].
    shape)
print(data["removed_columns"])
print(data["feature_names"][:10])
```

В числовом конвейере пропуски заменяются медианой, затем выполняется стандартизация. Категориальный конвейер заполняет пропуски отдельной категорией и выполняет one-hot-кодирование. `ColumnTransformer` направляет соответствующие столбцы в каждый конвейер. Для признаков с целочисленными кодами сверьтесь со словарем: формат хранения не определяет содержательный смысл признака.

Маркер **NaN (англ. not a number)** встречается при пропуске числового значения или неопределенном результате вычисления. Проверьте результат: три матрицы должны иметь одинаковое число столбцов и не содержать NaN или бесконечностей. Пересечение `train_ids` и `validation_ids` должно быть пустым, а объекты валидации должны проходить только через `transform`. Число входов сети равно `data["Atr"].shape[1]`.

Шаг 3. Рассчитайте константные прогнозы

Вычислите среднюю и медианную цену обучающей части. Используйте каждое значение как прогноз для всех валидационных домов и рассчитайте MAE и RMSE.

```
baseline_table = pd.DataFrame(constant_baselines(data)).T
print(baseline_table)
```

Константный прогноз позволяет проверить, насколько полезна информация из признаков. Если нейросеть не улучшила результат, это необходимо показать и разобрать возможные причины.

Шаг 4. Постройте сеть B0

Входная размерность определяется подготовленной матрицей. Скрытый слой содержит 64 нейрона с ReLU, линейный выход предсказывает одно стандартизованное значение цены. Компиляция задает Adam со скоростью обучения 10^{-3}, функцию потерь MSE и наблюдаемую MAE.

```
from tensorflow import keras

keras.utils.set_random_seed(40)
d = data["Atr"].shape[1]
model = keras.Sequential([
    keras.Input(shape=(d,)),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.Dense(1, activation="linear"),
])
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    loss="mse",
    metrics=[keras.metrics.MeanAbsoluteError(name="mae")],
)
model.summary()
```

Рассчитайте число параметров этой сети вручную и сопоставьте его со сводкой модели. При изменении ширины или глубины повторите расчет, чтобы связать устройство сети с ее сложностью.

Шаг 5. Обучите сеть и сохраните историю

Передайте `Atr` и стандартизованную цель `ztr`. В аргументе `validation_data` явно укажите `Ava` и `zva`. Таблица `test_data` не содержит ответов и для этого аргумента не подходит.

```
history = model.fit(
    data["Atr"], data["ztr"],
    validation_data=(data["Ava"], data["zva"]),
    epochs=100, batch_size=32, shuffle=True, verbose=0,
)
```

Функция `run_experiment` выполняет ту же последовательность целиком: создает новую модель, обучает ее, возвращает прогноз к исходной шкале и рассчитывает метрики. Выберите собственную последовательность шагов или эту функцию для одного опыта, поскольку ее вызов после `model.fit` запускает обучение еще одной сети.

```
from lab2_utils import run_experiment, plot_diagnostics

result = run_experiment(
    data, hidden=(64,), epochs=100, batch_size=32,
    optimizer="adam", learning_rate=1e-3,
    loss="mse", seed=40, early_stop=False,
)
print(result["metrics"])
fig = plot_diagnostics(data, result)
```

Шаг 6. Оцените качество и разберите ошибки

Для всех конфигураций используйте одни и те же валидационные дома. MAE и RMSE итогового сравнения рассчитывайте после обратного преобразования цели. Постройте три графика:

1. MAE обучения и валидации по эпохам.
2. Предсказанная цена против фактической; прямая $\widehat{y} = y$ показывает положение идеальных прогнозов.
3. Остаток (англ. residual) $e = y - \widehat{y}$ против предсказанной цены с горизонтальной линией $e = 0$.

В подписях укажите модель, часть данных и единицы измерения. По двум последним графикам определите, где возникают крупные промахи, систематическое завышение или занижение цены и изменение разброса ошибок.

Рассмотрите исходные характеристики пяти домов с наибольшей абсолютной ошибкой. Эти объекты остаются в валидации, даже если ухудшают метрику. Предполагаемую ошибку данных опишите как проверяемую гипотезу и укажите, какие сведения нужны для проверки.

Шаг 7. Получите и сохраните прогнозы

После выбора конфигурации вычислите прогноз для всех строк `test.csv`. В базовом варианте сохраняется модель, обученная на 80% размеченных строк; ее оценка получена на оставшихся 20%.

```
import numpy as np
from lab2_utils import save_experiment

preds = np.asarray(result["predictions"]).reshape(-1)
output = pd.DataFrame({
    "Id": data["prediction_ids"],
    "SalePrice": preds,
})
assert len(output) == len(test_data)
assert output["Id"].tolist() == test_data["Id"].tolist()
assert np.isfinite(output["SalePrice"]).all()
print("Negative prices:", int((output["SalePrice"] < 0).sum())
    )
save_experiment(data, result, "runs/baseline_01")
```

Сопоставление `Id` проверяет связь прогнозов с исходными домами. Отрицательная цена требует проверки обучения и обратного преобразования. Если вводится постобработка, обоснуйте ее по валидации и примените одинаково к прогнозам для валидации и `test.csv`. Файл `predictions_raw.csv` должен сохранять исходные значения.

После фиксации настроек допускается отдельное финальное переобучение на всем `train.csv`. В этом случае заново обучаются предобработка и сеть. Прежняя валидационная ошибка характеризует прежнюю модель; строки, вошедшие в финальное обучение, уже не дают независимой оценки новой модели.

2.10. Выполнение на PyTorch

При выборе PyTorch сохраните разбиение, предобработку, архитектуру, метрики и исследуемые факторы. Обычный интерфейс этой библиотеки требует явного цикла обучения. Вместо `model.compile()` и `model.fit()` выполняется следующая последовательность:

```
model.train()
for xb, yb in train_loader:
    optimizer.zero_grad()
    prediction = model(xb)
    loss = criterion(prediction, yb)
    loss.backward()
    optimizer.step()

model.eval()
with torch.no_grad():
    val_prediction = model(X_val_tensor)
```

В режиме обучения обнуляются предыдущие градиенты, вычисляется прогноз и loss, затем производные и обновление весов. Для валидации `model.eval()`

переключает поведение слоев, а `torch.no_grad()` отключает запись операций для вычисления градиентов. Это разные действия, поэтому выполняются оба.

Загрузчик, модель, оптимизатор и тензоры в этом фрагменте нужно создать самостоятельно. Прогноз и цель должны иметь одинаковую форму $(b, 1)$: сочетание $(b, 1)$ и $(b,)$ допускает нежелательное расширение размерностей при вычислении loss. Используйте `MSELoss` и `L1Loss` для соответствующих опытов. При одном seed TensorFlow и PyTorch могут получить разные веса и метрики; результаты сравниваются с учетом соблюдения общего протокола.

2.11. Исследование гиперпараметров

Сформулируйте ожидаемое влияние каждого изменения до запуска. В каждом сравнении изменяется один фактор относительно B0. Создавайте новую сеть и новый оптимизатор: при продолжении обучения в результат вмешивается предыстория весов.

> **Условия основных опытов**
>
> Разделение 80/20; seed 40; одинаковая подготовка признаков и цели; ReLU в скрытых слоях; линейный выход; скорость обучения 10^{-3}. Основные опыты проводятся без раннего останова.

План включает десять конфигураций. Преподаватель может скорректировать бюджет до начала занятия; принятые условия фиксируются до запусков.

Таблица 5: План сравнения. В столбце «слои» указаны ширины скрытых слоев.

Опыт	Слои	Эпохи	Batch	Оптимизатор	Loss
B0	64	100	32	Adam	MSE
W1	32	100	32	Adam	MSE
W2	128	100	32	Adam	MSE
D1	64, 32	100	32	Adam	MSE
E1	64	30	32	Adam	MSE
E2	64	200	32	Adam	MSE
P1	64	100	16	Adam	MSE
P2	64	100	64	Adam	MSE
O1	64	100	32	SGD	MSE
L1	64	100	32	Adam	MAE

Для каждой группы сравнений ответьте на свой вопрос:

1. **Ширина: B0, W1, W2.** Как изменение числа параметров связано с ошибкой и разрывом между обучением и валидацией?
2. **Глубина: B0, D1.** Что добавляет второй скрытый слой и оправдывает ли изменение ошибки усложнение сети?
3. **Эпохи: E1, B0, E2.** На каком участке кривых сохраняется улучшение? Что происходит к концу каждого запуска?

4. **Batch: P1, B0, P2.** Рассчитайте число обновлений весов и свяжите его с результатом. Равное число эпох при разных batch дает разное число обновлений.
5. **Оптимизатор: B0, O1.** Как изменились кривые и итоговая ошибка при фиксированном шаге? Вывод относится к этим настройкам; для сравнения Adam и SGD с индивидуально подобранными скоростями обучения нужны отдельные опыты.
6. **Функция потерь: B0, L1.** Сравните MAE и RMSE в исходных единицах цены. Какие крупные ошибки изменились при переходе к MAE?

Журнал результатов

Для каждого запуска сохраните идентификатор, архитектуру, число параметров, заданные и выполненные эпохи, размер batch, оптимизатор, скорость обучения, функцию потерь, seed, время обучения, MAE, RMSE и при необходимости R^2. Конфигурация и рассчитанные метрики собираются в одну строку:

```
row = {
    "experiment": "B0",
    **result["config"],
    **result["metrics"],
}
pd.DataFrame([row]).to_csv("experiment_B0.csv", index=False)
```

Выбор конфигурации

Выберите конфигурацию по MAE валидации. Затем рассмотрите RMSE, сложность сети и графики ошибок: они помогают объяснить выбор. Малую разницу по одному запуску следует описать как наблюдение на данном разбиении и указать проверку, нужную для оценки устойчивости.

В `test.csv` нет правильных цен, поэтому его прогнозы не дают основания подбирать гиперпараметры. В работе не задан универсальный порог MAE или обязательная победа над другими алгоритмами; оценка зависит от корректности опыта и обоснованности выводов.

Дополнительные исследования

После основных опытов можно выполнить дополнительные исследования:

- включите ранний останов по MAE валидации и сравните с B0; укажите фактическую длину обучения и правило возврата весов;
- исследуйте влияние отдельной скорости обучения у SGD;
- сравните фиксированное удаление пяти признаков с сохранением их содержательно интерпретированных категорий;
- повторите две выбранные конфигурации с несколькими заранее заданными seed и покажите разброс на том же фиксированном разбиении.

В журнале дополнительных опытов явно укажите измененные условия, чтобы их результат можно было сопоставить с исходным планом.

Дополнительный опыт: усреднение прогнозов

Несколько моделей могут ошибаться по-разному. Если одна завысила цену, а другая занизила, усреднение их прогнозов может уменьшить ошибку. Если обе ошибаются в одну сторону, такое улучшение не гарантировано.

1. На одном сохраненном разбиении обучите три новые сети B0 с seed 40, 41 и 42. Предобработка и остальные настройки одинаковы.
2. Для каждого дома усредните три прогноза в исходных единицах цены. Проверьте, что порядок домов в массивах совпадает.
3. Сравните MAE и RMSE трех сетей и усреднения на той же валидации. Учтите время трех обучений; разберите по одному дому, для которого усреднение помогло и не помогло, если такие случаи есть.

В конце ноутбука работу этого опыта включает `RUN_OPTIONAL_ENSEMBLE = True`. Для сдачи достаточно таблицы результатов и краткого вывода. Разброс трех прогнозов сам по себе не задает гарантированного интервала цены.

2.12. Разбор ошибок при выполнении работы

Наблюдение	Возможная причина	Проверка
Разное число входных признаков	Кодировщики обучены независимо для разных таблиц	Сопоставить имена и порядок столбцов; использовать один обученный кодировщик.
Подозрительно малая ошибка	Цена попала в признаки или статистики получены до разделения	Проверить входы и строки, по которым выполнен `fit`.
Loss равен NaN	Остались NaN или бесконечности; обучение расходится	Проверить массивы и цель, затем масштаб и скорость обучения.
Прогнозы около нуля	Сохранена стандартизованная цель	Проверить обратное преобразование с `target_scaler`, обученным на y_{train}.
Крупные ошибки у дорогих домов	Редкие сочетания признаков или ошибка в части диапазона	Рассмотреть остатки и характеристики этих домов; сохранить их в оценке.
Отрицательные цены после обратного преобразования	Ошибка шкалы или поведение модели	Проверить преобразование; оценить изменение модели или постобработку на валидации.
Результат меняется при повторе	Изменились разбиение, инициализация или начальные веса	Сопоставить seed, версии, идентификаторы строк и создание новой модели.
Ошибка формы тензоров в PyTorch	Цель имеет форму $(b,)$, выход – $(b, 1)$	Привести обе формы к $(b, 1)$.

2.13. Отчет и защита работы № 2

Оформите отчет по разделу «Общие требования к отчету» и шаблону приложения 3.10. Для лабораторной № 2 представьте следующие материалы:

1. Описание выбранных признаков и предобработки: удаленные столбцы, заполнение пропусков, кодирование категорий и масштабирование. Укажите разбиение 80/20 с seed 40 и покажите, что статистики вычислены только по обучающей части.
2. Архитектуру B0, размерности слоев и расчет числа параметров.
3. Таблицу MAE и RMSE константных прогнозов по средней и медиане обучающих цен и всех десяти конфигураций основного плана. Объясните влияние исследованных факторов, выбор по MAE валидации и ограничения вывода.
4. Три графика: MAE обучения и валидации по эпохам, предсказанные цены против фактических, остатки против предсказанных цен. Разберите характеристики пяти домов с наибольшей абсолютной ошибкой.
5. Пример итоговой таблицы с полями `Id` и `SalePrice`, а также полный CSV-файл прогнозов. Состав и порядок `Id` должны совпадать с `test.csv`; исходные прогнозы сохраняются в `predictions_raw.csv`.

К ноутбуку и коду изменений приложите журнал запусков и файлы `history.csv`, `validation_predictions.csv`, `model.keras`, `preprocessing.joblib` и `manifest.json`. Комплект должен фиксировать исходные данные, идентификаторы разбиения, конфигурации, seed и версии библиотек; сведения об устройстве укажите в отчете. Если вводилась постобработка цен или финальное переобучение на всем `train.csv`, опишите их отдельно и укажите, к какой модели относится валидационная оценка.

Вопросы к защите

1. Почему для `SalePrice` решается задача регрессии? Какие сведения о доме доступны модели при прогнозировании?
2. Как используются обучающие и валидационные строки `train.csv` и дома из `test.csv`? Где выбирались гиперпараметры вашей модели?
3. Как независимое обучение кодировщиков на двух таблицах нарушает соответствие признаков? Приведите пример с категориями.
4. Что изменило масштабирование признаков и целевой цены? Как вы вернули прогноз к исходным единицам?
5. Что вычисляет полносвязный слой и сколько параметров в каждом слое вашей сети?
6. Почему изменение ширины скрытого слоя сохраняет входную размерность при той же предобработке?
7. Как выбор ReLU, sigmoid или tanh влияет на скрытый слой? Почему для данной цели используется линейный выход?
8. Сколько обновлений весов выполнено за эпоху и за весь опыт? Как на расчет повлиял `batch_size`?
9. Что измеряют MSE, MAE и RMSE и в каких единицах вы приведете каждую величину?
10. Почему значения функций потерь MSE и MAE нельзя сравнивать напрямую? Какие общие метрики вы использовали?
11. Как ширина, глубина, число эпох, batch и оптимизатор повлияли на результаты? Какие наблюдения поддерживают выводы?
12. Какие участки кривых дают основания подозревать переобучение? Что изменил бы ранний останов и как выбираются возвращаемые веса?
13. Почему выбор по одной валидации ограничивает вывод о качестве на будущих домах?
14. Какие свойства таблицы `Id, SalePrice` можно проверить без известных цен и какие сведения нужны для оценки ошибки?

Практическая работа № 3. Сверточные нейронные сети

3.1. Постановка задачи

Требуется классифицировать изображения пяти видов цветов: ромашки, одуванчики, розы, подсолнухи и тюльпаны. Используется набор Flowers Recognition с каталогами `daisy`, `dandelion`, `rose`, `sunflower`, `tulip`.

Сверточная нейронная сеть (англ. convolutional neural network, CNN) выделяет признаки локальных участков изображения. Сравнение включает три модели:

1. заданная сверточная сеть, обучаемая с нуля;
2. сеть с весами, заранее полученными на другом наборе данных, приспособленная к классификации цветов;
3. собственная архитектура студента, обучаемая с нуля.

Общее разбиение позволяет сопоставить качество на одних и тех же фотографиях. Реализация дана на PyTorch. Для TensorFlow/Keras потребуется перенести слои, режимы, предобработку и протокол целиком.

3.2. Изображение как тензор

Тензор (англ. tensor) – многомерный массив чисел. Цветное изображение содержит три **канала (англ. channels)**: отдельные таблицы интенсивности красного, зеленого и синего цветов. Отсюда обозначение **RGB (англ. red, green, blue)**. В PyTorch batch изображений имеет форму (B, C, H, W): размер batch, число каналов, высота, ширина. Для принятого входа это $(B, 3, 224, 224)$. `ImageFolder` читает метки из имен каталогов и формирует `class_to_idx`. Это отображение связывает числовые ответы сети с названиями цветов.

Преобразование в тензор переводит обычные 8-битные RGB-пиксели в значения от 0 до 1. Затем выполняется нормализация каналов:

$$z_{c,h,w} = \frac{x_{c,h,w} - \mu_c}{\sigma_c}, \qquad x_{c,h,w} = \sigma_c z_{c,h,w} + \mu_c. \tag{3.1}$$

Здесь используются $\mu = (0{,}485, 0{,}456, 0{,}406)$ и $\sigma = (0{,}229, 0{,}224, 0{,}225)$ – фиксированные константы нормализации ImageNet. Статистики по набору цветов для этой операции не вычисляются. Индекс c обозначает канал, h, w – положение пикселя. При визуализации обратная формула восстанавливает цвет; прямой `clip(0,1)` нормализованного тензора его искажает.

Все три модели получают `Resize((224,224))`, преобразование в тензор и одну нормализацию. Такое изменение размера может менять пропорции изображения. Рекомендованная предобработка весов ResNet18, `weights.transforms()`, включает изменение размера и центральную обрезку. Ее влияние можно проверить отдельным опытом, применив выбранные преобразования ко всем сравниваемым моделям.

Аугментация данных (англ. data augmentation) создает измененные варианты обучающих изображений, например зеркальные отражения. Модель встречает больше допустимых вариантов объекта, поэтому преобразование

должно сохранять его класс. В основном сравнении случайные преобразования выключены; их влияние можно исследовать отдельно. На валидации преобразования фиксированы.

3.3. Сверточный слой

Полносвязный слой рассматривает все входные координаты совместно. **Свертка (англ. convolution)** применяет один набор весов к небольшому участку входа, затем сдвигает его и повторяет вычисление. Этот массив весов называют **ядром свертки (англ. kernel)**. Здесь ядро означает веса фильтра; ядро Jupyter исполняет код ноутбука. Расстояние между соседними положениями ядра называют **шагом (англ. stride)**. Для шага 1 и без дополнения границы один выходной канал вычисляется по правилу

$$a_{o,h,w} = b_o + \sum_{c=1}^{C_{in}} \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} W_{o,c,u,v}\, x_{c,h+u,w+v}. \tag{3.2}$$

Здесь o – выходной канал, c – входной; u, v перебирают элементы ядра размера $k_h \times k_w$. Веса W и смещение b_o обучаются. В `Conv2d` реализуется взаимная корреляция: ядро в этой записи не переворачивается. В нейронных сетях операцию традиционно называют сверткой. Полученные **карты признаков (англ. feature maps)** – таблицы откликов фильтров в разных участках изображения – проходят через нелинейность, например ReLU. Обучение определяет, какие локальные сочетания полезны для классификации.

> **Пример вычисления свертки**
>
> Для одной карты $X = \left[\begin{smallmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{smallmatrix}\right]$, ядра $K = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}\right]$, нулевого смещения и шага 1 без дополнения границы получается $\left[\begin{smallmatrix} 11 & 14 \\ 20 & 23 \end{smallmatrix}\right]$. Например, левый верхний элемент равен $1 + 2 \cdot 5 = 11$. Найдите остальные элементы вручную.

Дополнение границы (англ. padding) добавляет значения по краям входа, в нашей сети – нули. **Разрежение ядра (англ. dilation)** задает расстояние между его соседними элементами при наложении. Для размера входа H, ядра k, шага s, дополнения p и разрежения d высота выхода равна

$$H_{out} = \left\lfloor \frac{H + 2p - d(k-1) - 1}{s} \right\rfloor + 1. \tag{3.3}$$

Знаки $\lfloor x \rfloor$ обозначают округление вниз до ближайшего целого: например, $\lfloor 2{,}5 \rfloor = 2$. Для ширины действует такая же формула. В этой работе $d = 1$. При $k = 3$, $s = 1$, $p = 1$ размер сохраняется. Пусть C_{in} и C_{out} – число входных и выходных каналов. Для обычной свертки без групп со смещением число параметров

$$P = C_{out}(C_{in} k_h k_w + 1). \tag{3.4}$$

Первый слой $3 \to 32$ с ядром 3×3 содержит $32(3 \cdot 3 \cdot 3 + 1) = 896$ параметров. Эти веса используются во всех положениях ядра, поэтому их число не растет с площадью изображения.

3.4. Пулинг и устройство исходной сети

Пулинг (англ. pooling) объединяет значения в небольшом окне, уменьшая число пространственных координат. `MaxPool2d(2,2)` выбирает максимум в каждом окне 2×2 и уменьшает высоту и ширину вдвое для четного входа. Он не обучает веса. Пулинг уменьшает пространственную детализацию и стоимость следующих вычислений; это не означает полной устойчивости классификатора к любому сдвигу или повороту. Перед полносвязными слоями операция **развертывания (англ. flatten)** располагает значения карт в одном векторе, сохраняя их число; в коде ей соответствует `Flatten`. В таблице Conv и Linear обозначают соответственно сверточный и полносвязный слои.

Таблица 7: Размерности исходной CNN при входе 224×224

Последовательность операций	Форма без размера batch
Вход RGB	$3 \times 224 \times 224$
Conv $3 \to 32$, ReLU, Conv $32 \to 64$, ReLU, MaxPool	$64 \times 112 \times 112$
Conv $64 \to 128$, ReLU, Conv $128 \to 128$, ReLU, MaxPool	$128 \times 56 \times 56$
Conv $128 \to 256$, ReLU, Conv $256 \to 256$, ReLU, MaxPool	$256 \times 28 \times 28$
Flatten	200704
Linear, ReLU; Linear, ReLU; Linear	$1024 \to 512 \to 5$

Все свертки в таблице имеют ядро 3, шаг 1 и дополнение границы 1. Эта архитектура реализована функцией `original_cnn()`.

Формат `float32` хранит 32-битные числа с плавающей запятой: на одно число приходится 4 байта.

> **Почему нужно заранее оценить память**
>
> Слой $200704 \to 1024$ содержит $205\,521\,920$ параметров со смещением. Вся исходная сеть – $207\,175\,365$. Только веса float32 занимают около 0,83 ГБ. Веса, градиенты и два состояния Adam суммарно требуют около 3,31 ГБ, еще без активаций и временных буферов. Это расчет по архитектуре, а не измерение пикового расхода памяти.

Слой `AdaptiveAvgPool2d(1)` усредняет каждую карту, превращая $C \times H \times W$ в $C \times 1 \times 1$. Последующие слои получают один средний отклик на канал и предсказывают пять классов. Сокращение их входа уменьшает число параметров; влияние на качество проверяется экспериментом.

3.5. Логиты, перекрестная энтропия и обучение

Последний слой выдает пять логитов $z_1, \ldots, z_5$. Вероятности классов вычисляются по формуле softmax (2.10) при $K = 5$. **Перекрестная энтропия (англ. cross-entropy)** штрафует модель за низкую вероятность правильного класса. Для batch из B объектов с правильными классами y_i значение loss зависит от вероятностей этих классов:

$$L = -\frac{1}{B} \sum_{i=1}^{B} \log p_{i,y_i}. \tag{3.5}$$

`CrossEntropyLoss` принимает исходные логиты формы $(B, 5)$ и целочисленные индексы классов от 0 до 4 формы $(B,)$ с типом `torch.long`. Она включает нужное преобразование логитов, поэтому отдельный Softmax перед ней не нужен. Для выбора класса достаточно `logits.argmax(dim=1)`: наибольший логит и наибольшая вероятность имеют один индекс. Значение softmax само по себе не подтверждает надежность прогноза.

Итерация обучения включает обнуление градиентов, прямой проход, вычисление loss, обратный проход и шаг оптимизатора. При проверке выполняются прямой проход и расчет метрик. `model.train()` включает режим обучения слоев, а `model.eval()` – режим применения модели; конкретное поведение зависит от слоя. Эти режимы управляют поведением сети, а роли обучающей и валидационной частей задает разбиение данных. Вызов `torch.no_grad()` отключает построение графа для расчета градиентов. Для проверки нужны `eval()` и `no_grad()`.

Метрики считают заново для каждой эпохи и отдельно для обучения и валидации. При усреднении loss каждый batch учитывается по числу изображений n_b:

$$L_{epoch} = \frac{\sum_b n_b L_b}{\sum_b n_b}. \tag{3.6}$$

Например, для batch из 32 и 8 изображений нельзя брать простое среднее двух значений loss. Accuracy считают по всем правильным ответам фазы, macro-F1 – по всем ее меткам и прогнозам, а не как невзвешенное среднее F1 batch.

3.6. Перенос обучения и заморозка

Перенос обучения (англ. transfer learning) использует в новой задаче веса, полученные при решении другой. При **предобучении (англ. pretraining)** сеть заранее получает веса на другом наборе данных. Основную часть сети, преобразующую изображение в признаки, называют **извлекателем признаков (англ. backbone)**. Завершающий блок переводит признаки в ответы нашей задачи; это **голова классификации (англ. head)**. В этой работе используется перенос с фиксированным извлекателем и обучением новой головы. При **дообучении (англ. fine-tuning)** дополнительно изменяют часть предобученных слоев.

В 2019 году команда Pinterest описала обучаемые векторные представления изображений (англ. embeddings), применявшиеся в визуальном поиске и

рекомендациях [7]. Такое представление можно использовать в разных задачах; в нашей работе голова классификации переводит признаки изображения в пять логитов, по которым выбирается класс.

Пример использует ResNet18 с явно указанными весами `IMAGENET1K_V1`. Основой сети служат остаточные блоки: к преобразованию входа добавляется путь, передающий сам вход. Если размеры совпадают, блок имеет вид $h_{next} = \mathrm{ReLU}(F(h) + h)$; при изменении размеров применяется согласующее преобразование короткого пути. Четыре группы блоков используют 64, 128, 256 и 512 каналов; после глобального усреднения заменяется слой `fc` на $512 \to 5$ [8].

Заморозка (англ. freezing) фиксирует выбранные веса на время обучения. Установка `requires_grad=False` исключает параметры из расчета их градиентов. Новая голова создается после заморозки и остается обучаемой; ее параметры включаются в оптимизатор.

Нормализация по batch (англ. batch normalization, BatchNorm) в режиме `train` нормализует промежуточные значения по статистикам batch и обновляет накопленные оценки средних и дисперсий в служебных значениях – **буферах (англ. buffers)**. В режиме `eval` слой использует сохраненные оценки. Буферы меняются при обучении даже с замороженными параметрами. Поэтому для полностью фиксированного извлекателя backbone работает в `eval`, а новая голова – в `train`. Общий вызов `model.train()` переключил бы и backbone в режим обучения.

Веса ResNet получены на внешних данных. При сравнении с обучением с нуля это дополнительный ресурс: результат характеризует архитектуру вместе с выбранным способом получения весов.

3.7. Подготовка изображений и разбиение

Шаг 1. Изучите набор и проверьте ресурсы

Выведите классы, число изображений каждого класса и несколько исходных примеров. Откройте одно изображение и сопоставьте его с меткой. Проверьте читаемость файлов, размеры, цветовые каналы и дубликаты: состав набора зависит от версии выгрузки. Выберите устройство и запишите его в отчете.

```
import torch
from lab3_utils import prepare_flowers, seed_everything
from lab3_utils import original_cnn, transfer_resnet18,
    compact_example
from lab3_utils import fit_flowers, plot_learning,
    parameter_counts
seed_everything(0)
device = "cuda" if torch.cuda.is_available() else "cpu"
print(device)
```

Шаг 2. Зафиксируйте одну валидацию

Целевой размер валидации – 1000 изображений. Точные копии файлов объединяются по SHA-256 и целиком попадают в одну часть, поэтому фактический размер может отличаться от 1000. `split.csv` сохраняет итоговое распределение. Выведите пропорции классов и проверьте наличие всех пяти классов в обеих частях: групповое разделение само по себе этого не гарантирует.

```
data = prepare_flowers("data", validation_target=1000,
                       seed=0, image_size=224, augment=False)
print(data["class_to_idx"])
print(data["inventory"].groupby(["split", "target"]).size())
print(data["config"])
```

Хеши находят побайтовые копии. После повторного сжатия, обрезки или изменения размера снимки одной сцены могут иметь разные хеши. Если такие серии известны, объедините их до разделения. Совпадающие файлы с разными метками требуют проверки разметки; причину исправления укажите в отчете.

Функция создает отдельные объекты `ImageFolder` для обучения и валидации с одним списком файлов и разными подмножествами. Это позволяет в дополнительном опыте применять случайное отражение только к обучению. Если два `Subset` используют общий объект, замена его `transform` изменит обе части одновременно.

3.8. Обучение и исследование архитектур

Шаг 3. Разберите и обучите заданную CNN

Выпишите размеры тензоров после сверток и пулинга, проверьте вход первого Linear и число параметров. Оцените память по расчету из теоретического раздела. Создайте новую модель, затем оптимизатор. Изучите `fit_flowers` в `lab3_utils.py`: найдите сброс метрик, переключение режима и выбор сохраненных весов. Параметры основного запуска заданы явно:

```
seed_everything(0)
model = original_cnn()
print(parameter_counts(model))
result_original = fit_flowers(
    model, data, "runs/lab3_original_01", epochs=10,
    batch_size=32, learning_rate=1e-3, seed=0, device=device
)
```

Перед полным запуском выполните прямой и обратный проход на одном batch, проверьте форму выхода. Такой запуск проверяет совместимость размеров и достаточность памяти. Если памяти недостаточно, уменьшите batch и зафиксируйте общий протокол для всех трех моделей. Уменьшение разрешения у исходной сети потребует изменения первого Linear; измененную архитектуру

укажите отдельно. Ошибку памяти зафиксируйте как неудавшийся запуск без метрики качества.

Шаг 4. Примените перенос обучения

Выберите предобученную архитектуру, прочитайте ее описание и определите слой классификатора. Для ResNet18 заморозьте старые параметры, затем создайте новую голову. В таком порядке ее параметры сохраняют разрешение на вычисление градиентов:

```
from torchvision import models
from torch import nn
weights = models.ResNet18_Weights.IMAGENET1K_V1
model = models.resnet18(weights=weights)
for p in model.parameters():
    p.requires_grad = False
model.fc = nn.Linear(model.fc.in_features, 5)
model.frozen_backbone = True
for name, p in model.named_parameters():
    if p.requires_grad:
        print(name, p.shape)
```

Вывод должен содержать только `fc.weight` и `fc.bias`. Для головы $512 \to 5$ обучается $512 \cdot 5 + 5 = 2565$ параметров. При первом использовании веса могут загружаться из сети. Значение `weights=None` создает случайную инициализацию; для переноса нужны предобученные веса.

Функция `transfer_resnet18()` выполняет эту настройку и сохраняет обозначение весов в метаданных. Обучите новую модель в отдельном каталоге:

```
seed_everything(0)
model = transfer_resnet18(pretrained=True)
result_transfer = fit_flowers(
    model, data, "runs/lab3_transfer_01", epochs=10,
    batch_size=32, learning_rate=1e-3, seed=0, device=device
)
```

Проверьте, что после шага обучения параметры извлекателя признаков и накопленные статистики BatchNorm сохранились. Для дополнительного опыта с дообучением укажите размороженные слои, режим BatchNorm и новый оптимизатор: эти настройки определяют, какая часть сети меняется.

Шаг 5. Создайте собственную архитектуру

Выберите проверяемую гипотезу и предложите соответствующее изменение: размер головы, ширину или глубину блоков, регуляризацию, способ агрегирования карт. Нарисуйте последовательность слоев, рассчитайте формы и

число параметров. Выход должен содержать пять логитов. Пример использует глобальное усреднение:

```
student_model = nn.Sequential(
    nn.Conv2d(3, 16, 3, padding=1), nn.ReLU(), nn.MaxPool2d(2),
    nn.Conv2d(16, 32, 3, padding=1), nn.ReLU(), nn.MaxPool2d(2),
    nn.Conv2d(32, 64, 3, padding=1), nn.ReLU(),
    nn.AdaptiveAvgPool2d(1), nn.Flatten(), nn.Linear(64, 5)
)
```

Внесите как минимум одно обоснованное структурное изменение относительно примера `compact_example()` и объясните ожидаемый эффект. Зафиксируйте seed *до* создания слоев, чтобы он определял начальные веса. Обучите свой вариант с тем же разбиением и основным бюджетом.

3.9. Сравнение, визуализация и сохранение

Шаг 6. Соберите сопоставимые результаты

Основной протокол: RGB 224×224, фиксированная нормализация, без случайной аугментации, seed 0, Adam с шагом 10^{-3}, batch 32, десять эпох. Для каждой модели создаются новые загрузчики с тем же начальным seed. Выбирается эпоха с наибольшим macro-F1 валидации; при равенстве сохраняется более ранняя. Укажите предобучение ResNet18 как дополнительный ресурс.

Таблица 8: Что должно быть в итоговой таблице CNN

Группа полей	Содержание
Модель	Имя, схема, число всех и обучаемых параметров; исходные веса
Обучение	Разбиение, seed, эпохи, batch, оптимизатор, шаг, размер входа, аугментация
Результат	Лучшая эпоха, loss, accuracy и macro-F1 одной валидации
Ресурсы	Устройство, измеренное время с указанием границ измерения; при измерении – пиковая память
Вывод	Какие классы и случаи улучшились или ухудшились; ограничения сравнения

В `fit_flowers` время охватывает обучение, валидацию и запись контрольных точек. Первоначальная загрузка предобученных весов и подготовка набора выполняются вне этого измерения. Сопоставляйте время за одинаковые этапы; одинаковое число эпох у разных сетей может требовать разного числа операций.

Шаг 7. Постройте графики и исследуйте прогнозы

Для каждой модели покажите loss и macro-F1 по эпохам на обучении и валидации, а затем общую диаграмму результатов трех архитектур. Сохраните матрицы ошибок и отчеты по пяти классам. Используйте одинаковые оси и порядок меток, подпишите доли или проценты.

```
fig = plot_learning(result_transfer)
fig.savefig("runs/lab3_transfer_01/curves.pdf", bbox_inches="
    tight")
```

Покажите девять заранее выбранных изображений валидации либо все, если их меньше. Для каждого укажите истинный и предсказанный класс, восстановите цвета. Дополнительно разберите несколько ошибок. Для прогноза одного нормализованного изображения выполните:

```
from lab3_utils import denormalize
import matplotlib.pyplot as plt
index = int(data["validation_indices"][0])
image, label = data["validation_dataset"][index]
model = result_transfer["model"]
model.eval()
with torch.no_grad():
    logits = model(image.unsqueeze(0).to(device))
predicted = int(logits.argmax(1).item())
plt.imshow(denormalize(image).permute(1, 2, 0))
plt.title(f"True: {data['classes'][label]}; predicted: {data['
    classes'][predicted]}")
plt.axis("off")
```

Шаг 8. Сохраните воспроизводимый комплект

Каждая архитектура получает свой каталог. Основные файлы:

- `split.csv` – список изображений и разбиение;
- `history.csv` – история обучения по эпохам;
- `best_weights.pth` – выбранные веса;
- `manifest.json` – параметры и результаты опыта;
- `architecture.txt` – текстовое описание модели.

Также сохраняются прогнозы, отчет и матрица ошибок. Приложите исходный код собственной архитектуры. Для восстановления вместе с весами нужны совместимое определение модели, те же преобразования и тот же словарь классов.

3.10. Контрольные вопросы

1. Чем CNN отличается от полносвязной сети по связям, параметрам и работе с пространством?
2. Что задают kernel size, stride, padding, входные и выходные каналы? Рассчитайте размер выхода одного слоя.
3. Как работают свертка и pooling в исходной сети? Какие из них имеют обучаемые параметры?
4. Почему первый Linear исходной архитектуры требует так много памяти?
5. Что такое логиты? Почему перед CrossEntropyLoss не добавляется Softmax?
6. Зачем нужны одновременно eval и отключение градиентов?
7. Что такое предобучение, transfer learning и fine-tuning? Откуда взяты веса выбранной модели?
8. Как устроена выбранная предобученная архитектура? Для ResNet объясните короткий путь и сложение в блоке.
9. Что делает requires_grad? Почему заморозка параметров не равна заморозке статистик BatchNorm?
10. Какое изменение вы внесли в собственную сеть и какую гипотезу проверяли?
11. Почему нельзя накапливать счетчики разных эпох или усреднять F1 batch как F1 всего набора?
12. Как корректно показать нормализованное изображение? Почему clip без обратной нормализации неверен?
13. Что можно и чего нельзя заключить из сравнения с предобученной сетью?

Общие требования к отчету

Структура отчета

Используйте шаблон приложения 3.10. В основном тексте оставьте постановку задачи, условия сравнения, результаты и разбор ошибок. Ноутбук, полные таблицы прогнозов и файлы модели приложите отдельно. Названия опытов в отчете должны совпадать с именами сохраненных запусков.

Критерии оценивания

Критерий	Наблюдаемый признак выполнения
Данные	Известны признаки и цель; нет утечки, перепутанных выборок и неверных меток.
Модель	Студент объясняет алгоритм, размерности, функцию потерь и изменяемые настройки.
Эксперимент	Сравнения воспроизводимы; сохранены разбиения, seed, версии и результаты.
Анализ	Показаны ошибки и ограничения; выводы следуют из полученных результатов.
Самостоятельность	Есть обоснованный выбор, исследование параметров и собственная архитектура в работе № 3.

Баллы, минимальные пороги и правила пересдачи задает рабочая программа дисциплины. Качество прогноза оценивают вместе с корректностью эксперимента: утечка данных может повысить метрику, но делает вывод ненадежным. Неудачный вариант модели также включите в сравнение и объясните по результатам диагностики.

Обсуждение результатов на защите

Подготовьтесь объяснить путь одного объекта от исходных признаков до прогноза, выбранную метрику и условия сравнения моделей. Для изменения гиперпараметра сопоставьте ожидаемый эффект с полученным. Расхождение между ними может указывать на переобучение, ограничения данных или ошибку реализации; различить эти причины помогают графики и отдельные прогнозы.

Список литературы

[1] Николенко С. И., Кадурин А. А., Архангельская Е. О. Глубокое обучение. – СПб.: Питер, 2026. – 480 с. https://www.piter.com/collection/A31900/product/glubokoe-obuchenie-2.

[2] Babushkin V., Kravchenko A. Machine Learning System Design: With End-to-End Examples. Manning, 2025. https://www.manning.com/books/machine-learning-system-design.

[3] Николенко С. И. Машинное обучение: основы. – СПб.: Питер, 2026. – 608 с. https://www.piter.com/collection/biblioteka-programmista/product/mashinnoe-obuchenie-osnovy.

[4] Mamaev A. Flowers Recognition. Набор изображений из исходного задания. https://www.kaggle.com/datasets/alxmamaev/flowers-recognition.

[5] Reyes-Ortiz J., Anguita D., Ghio A., Oneto L., Parra X. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2013. https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones.

[6] Del Balso M., Hermann J. Meet Michelangelo: Uber's Machine Learning Platform. Uber Engineering, 5 September 2017. https://www.uber.com/us/en/blog/michelangelo-machine-learning-platform/.

[7] Zhai A., Wu H.-Y., Tzeng E., Park D. H., Rosenberg C. Learning a Unified Embedding for Visual Search at Pinterest. KDD, 2019. https://arxiv.org/abs/1908.01707.

[8] He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. CVPR, 2016. https://arxiv.org/abs/1512.03385.

Приложение А. Материалы эксперимента

Каждому запуску назначьте отдельную папку. Ниже приведен пример для работы № 2:

```
runs/B0_01/
    manifest.json
    history.csv
    validation_predictions.csv
    predictions_raw.csv
    model.keras
    preprocessing.joblib
    diagnostics.pdf
```

`manifest.json` описывает условия запуска: конфигурацию, разбиение, версии библиотек, контрольные суммы данных, подготовленные признаки и метрики валидации. По `history.csv` можно восстановить ход обучения, а `validation_predictions.csv` содержит пары истинных и предсказанных цен для разбора ошибок. Сохраненная модель применяется вместе с ее `preprocessing.joblib`: этот файл определяет заполнение пропусков, кодирование категорий и масштабирование входа и цели.

Графики добавляются после сохранения опыта:

```
fig.savefig("runs/B0_01/diagnostics.pdf", bbox_inches="tight")
```

Для названий файлов придерживайтесь одного идентификатора опыта: в приведенной строке папка `runs/B0_01` должна совпасть с аргументом вашего вызова `save_experiment`. Не открывайте файлы сериализованной предобработки из неизвестных источников; этот комплект предназначен для сохраненных вами опытов.

Пример вывода по сравнению моделей

Условный пример: при увеличении ширины скрытого слоя с 64 до 128 нейронов RMSE на обучении снизилась с 25 до 20 тысяч денежных единиц, а на валидации выросла с 30 до 33 тысяч. Разбиение и остальные настройки оставались фиксированными. Такая картина согласуется с переобучением: более широкая сеть лучше описывает обучающие объекты, но ее прогноз для новых объектов ухудшается. Один запуск еще не показывает устойчивость эффекта; для этого нужно повторить сравнение с другими начальными весами.

В собственном выводе разделяйте наблюдение, его возможное объяснение и ограничение эксперимента. Числа этого примера не являются результатами на наборе курса.

Что сохраняется в других работах

В работе № 1 сохраняются таблица сравнения валидации, тестовые прогнозы, classification report, обученный Pipeline и описание группового разбиения.

В работе № 3 – список изображений с метками и разбиением, история каждой эпохи, выбранные веса, описание архитектуры, словарь классов, метрики и прогнозы.

Для повторения собственной CNN сохраните ее исходный код, преобразование входного изображения и словарь классов вместе с весами.

Словарь основных обозначений

Символ	Значение
$x_i, y_i, \widehat{y}_i$	Признаки, истинный ответ и прогноз для объекта i.
d, n	Число признаков и число объектов в текущем наборе.
θ	Обучаемые параметры модели.
B	Размер batch; фактический последний batch может быть меньше.
η	Скорость обучения.
C_{in}, C_{out}	Число входных и выходных каналов свертки.
H, W	Высота и ширина карты признаков.
TP, FP, FN	Верно найденные объекты класса, ложные срабатывания и пропуски.

В работе № 1 параметр C без индекса обозначает гиперпараметр LinearSVC; в матрице C_{ij} индексы обозначают пару истинного и предсказанного классов. Смысл определяется контекстом, а не одним символом.

Приложение Б. Шаблон отчета

Требования к оформлению титульного листа, объему и сроку сдачи сообщает преподаватель.

Сведения о работе

Дисциплина: _____
Лабораторная работа № ____. Название: _____
Выполнил(а): _____ Группа: _____
ИТМО ID: _____ Проверил(а): _____
Дата: _____ Версия материалов: _____

1. Цель и данные

Сформулируйте цель своими словами. Укажите источник и версию файлов, объект наблюдения, признаки и целевой ответ. Приведите размер набора и проверки его состава. Объясните, какие данные доступны при применении модели.

2. Условия эксперимента

Опишите разбиение и способ исключения утечки. Укажите среду, seed, модель, основную метрику выбора и бюджет. Для каждого опыта сформулируйте ожидаемый эффект изменения параметра до запуска. Дополнительные опыты явно отделяйте от исходного плана.

3. Результаты и разбор ошибок

Для сравнения опытов используйте следующую структуру таблицы.

Опыт	Что изменено	Метрика на указанной части	Наблюдение
Исходный	Конфигурация модели	Значение и единицы	Что видно по результату
Сравнение	Один фактор либо явно заданная комбинация	Та же метрика и часть	Изменение относительно исходного

Сошлитесь на диагностические графики и разберите конкретные ошибки. Для неудачного запуска запишите причину остановки; метрика такого опыта остается неопределенной.

4. Выводы и приложение к отчету

Сформулируйте подтвержденный результат и ограничения вывода. Укажите пути к ноутбуку и сохраненным материалам опытов; состав файлов описан в приложении А.

Приложение В. Глоссарий

Термины расположены по алфавиту; термины с латинским написанием вынесены в конец. Номер страницы ведет к объяснению в тексте.

Термин и английское название	Краткое определение	С.
Ансамбль (англ. ensemble)	Несколько моделей, прогнозы которых объединяются в один ответ.	16
Аугментация данных (англ. data augmentation)	Преобразование обучающих примеров, сохраняющее их правильный ответ; например, поворот изображения цветка.	40
Базовая модель (англ. baseline)	Исходный вариант для сравнения: например, прогноз одной медианной цены для всех домов.	8
Бинарные индикаторы (англ. one-hot encoding)	Кодирование категории отдельным столбцом: 1 при ее наличии, 0 в остальных случаях.	23
Валидационная выборка (англ. validation set)	Отложенные объекты, по которым сравнивают конфигурации и выбирают настройки модели.	9
Гиперпараметры (англ. hyperparameters)	Настройки, задаваемые исследователем, например число соседей, ширина слоя или скорость обучения.	7
Голова сети (англ. head)	Выходная часть сети, которая превращает извлеченные признаки в ответ для конкретной задачи.	43
Градиент (англ. gradient)	Вектор производных функции по параметрам; указывает направление ее наиболее быстрого роста вблизи текущей точки.	27
Доля верных ответов (англ. accuracy)	Число правильных предсказаний класса, деленное на число всех оцениваемых объектов.	17
Дообучение (англ. fine-tuning)	Изменение весов предобученной сети при обучении на данных новой задачи.	43
Извлекатель признаков (англ. backbone)	Часть сети, преобразующая исходное изображение в признаки для выходной части.	43
Инференс (англ. inference)	Получение прогнозов с помощью уже обученной модели.	8

Продолжение на следующей странице

Термин и английское название	Краткое определение	С.
Итоговый тест (англ. test set)	Отдельные объекты с известными ответами для оценки уже выбранного решения.	9
Итерация (англ. iteration)	Один шаг обновления параметров по batch; эпоха обычно состоит из нескольких итераций.	28
Классификация (англ. classification)	Предсказание одной из заданных категорий, например вида цветка.	5
Конвейер (англ. pipeline)	Последовательность преобразований данных и применения модели; в scikit-learn ее задает класс `Pipeline`.	15
Логиты (англ. logits)	Числовые оценки классов на выходе сети до преобразования в вероятности.	26
Масштабирование признаков (англ. feature scaling)	Приведение значений к сопоставимым масштабам. Стандартизация вычитает среднее и делит на стандартное отклонение.	14
Матрица ошибок (англ. confusion matrix)	Таблица количества объектов для каждой пары правильного и предсказанного классов.	16
Машинное обучение (англ. machine learning, ML)	Построение правил распознавания или прогнозирования по примерам.	4
Медиана (англ. median)	Центральное значение упорядоченного ряда; при четном числе элементов – среднее двух центральных.	13
Метод ближайших соседей (англ. k-nearest neighbors, KNN)	Классификатор, определяющий ответ по меткам ближайших обучающих объектов.	15
Метод опорных векторов (англ. support vector machine, SVM)	Метод, строящий разделяющую границу с учетом зазора до классов и штрафа за нарушения.	15
Метрика качества (англ. metric)	Числовая оценка прогнозов по известным ответам; например, MAE или macro-F1.	6
Многослойный персептрон (англ. multilayer perceptron, MLP)	Полносвязная сеть с одним или несколькими скрытыми слоями.	24
Модель (англ. model)	Настраиваемое правило, которое по входным данным вычисляет прогноз.	5

Продолжение на следующей странице

Термин и английское название	Краткое определение	С.
Набор данных (англ. dataset)	Совокупность объектов с их характеристиками и, при наличии, правильными ответами.	5
Начальное значение генератора (англ. random seed)	Число, задающее начальное состояние генератора случайных чисел; помогает повторять разбиение и инициализацию.	10
Недообучение (англ. underfitting)	Модель не освоила полезную зависимость; качество недостаточно и на обучении, и на проверке.	8
Нормализация по batch (англ. batch normalization, BatchNorm)	Слой, преобразующий активации; при обучении использует статистики batch, при оценке – накопленные статистики.	44
Обратное распространение ошибки (англ. backpropagation)	Вычисление производных функции потерь через слои сети от выхода к входу по цепному правилу.	28
Обучающая выборка (англ. training set, train)	Объекты, по которым настраивают параметры модели и обучаемой предобработки.	9
Обучение с учителем (англ. supervised learning)	Обучение по примерам, для которых известны правильные ответы.	5
Оптимизатор (англ. optimizer)	Алгоритм обновления параметров модели; в работе № 2 сравниваются SGD и Adam.	28
Перекрестная энтропия (англ. cross-entropy)	Функция потерь для классификации, растущая, когда модель приписывает правильному классу малую вероятность.	43
Перенос обучения (англ. transfer learning)	Использование представлений или весов, полученных на другой задаче, для новой задачи.	43
Переобучение (англ. overfitting)	Подстройка под особенности обучающих примеров, ухудшающая перенос результата на новые данные.	8
Полнота (англ. recall)	Доля объектов выбранного класса, которые модель правильно отнесла к этому классу.	17
Предобработка (англ. preprocessing)	Подготовка исходных данных к входу модели: заполнение пропусков, кодирование категорий, изменение масштаба.	8
Признак (англ. feature)	Входная характеристика объекта: например, площадь дома или измерение датчика.	5

Продолжение на следующей странице

Термин и английское название	Краткое определение	С.
Пулинг (англ. pooling)	Агрегирование соседних значений карты признаков, обычно с уменьшением ее пространственного размера.	42
Ранний останов (англ. early stopping)	Прекращение обучения после заданного числа эпох без улучшения валидационного показателя.	29
Регрессия (англ. regression)	Предсказание числового значения, например цены дома.	5
Регуляризация (англ. regularization)	Ограничение сложности модели; например, штраф за большие веса.	15
Свертка (англ. convolution)	Применение одного набора весов к локальным участкам входа для построения карты признаков.	41
Сверточная сеть (англ. convolutional neural network, CNN)	Сеть со сверточными слоями, извлекающая локальные признаки изображений.	40
Скорость обучения (англ. learning rate)	Настройка, регулирующая величину шага обновления параметров.	28
Случайное отключение (англ. dropout)	Отключение части выходов слоя при обучении; при прогнозировании оно прекращается.	29
Случайный лес (англ. random forest)	Ансамбль деревьев решений, обученных с использованием случайности и объединяющих прогнозы.	16
Тензор (англ. tensor)	Многомерный массив; batch RGB-изображений задается четырьмя осями: объекты, каналы, высота, ширина.	40
Точность положительных прогнозов (англ. precision)	Доля правильных ответов среди объектов, которые модель отнесла к выбранному классу.	16
Утечка данных (англ. data leakage)	Использование при настройке модели информации, которая недоступна при ее дальнейшем применении.	9
Функция активации (англ. activation function)	Преобразование выхода нейрона перед передачей следующему слою; нелинейность позволяет описывать сложные зависимости.	24
Функция потерь (англ. loss function, loss)	Численная ошибка, которую алгоритм уменьшает при обучении.	26

Продолжение на следующей странице

Термин и английское название	Краткое определение	С.
Целевой ответ (англ. target)	Значение, которое требуется предсказать для объекта.	5
Эпоха (англ. epoch)	Один проход обучения по всей обучающей выборке; обычно включает несколько обновлений по batch.	28
Ядро свертки (англ. convolution kernel, filter)	Набор обучаемых весов, применяемый к каждому локальному участку входа.	41
Batch (англ. mini-batch)	Мини-выборка – группа обучающих объектов, используемая для одного обновления параметров сети.	28
ELU (англ. exponential linear unit)	Активация с линейной положительной и экспоненциальной отрицательной ветвями.	26
F1-мера (англ. F1 score)	Гармоническое среднее precision и recall; macro-F1 усредняет F1 классов с одинаковыми весами.	17
MAE (англ. mean absolute error)	Среднее абсолютных ошибок; измеряется в единицах целевого ответа.	6
MSE (англ. mean squared error)	Среднее квадратов ошибок; сильнее увеличивается при крупных промахах.	26
PReLU (англ. parametric rectified linear unit)	Модификация ReLU с обучаемым наклоном отрицательной ветви.	25
RMSE (англ. root mean squared error)	Корень из MSE; измеряется в единицах целевого ответа.	26
Softmax (англ. softmax)	Преобразование вектора оценок классов в вероятности с суммой 1.	26

Евстафьев Олег Александрович
Ким Станислав Александрович

Прикладной искусственный интеллект. Лабораторный практикум

Практикум

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н. Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А