МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

А. С. Степашкина

ЧИСЛЕННЫЕ МЕТОДЫ И МАШИННОЕ ОБУЧЕНИЕ В МЕТРОЛОГИИ

Учебное пособие

ГУАП
Санкт-Петербург
2021

УДК 006.91(073)
ББК 30.10я73
С79

Рецензенты:
кандидат технических наук, доцент *О. А. Москалюк;*
доктор технических наук, профессор *А. С. Коновалов*

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

Протокол № 5 от 24 августа 2021 г.

Степашкина, А. С.

С79 Численные методы и машинное обучение в метрологии: учеб. пособие / А. С. Степашкина. – СПб.: ГУАП, 2021. – 44 с.

ISBN 978-5-8088-1656-5

Посвящено применению численных методов для обработки экспериментальных результатов: аппроксимации функций, поиска локальных минимумов, поиска грубых промахов и др. Рассмотрены методы машинного обучения и метрики качества моделей.

Составлено в соответствии с программой дисциплины «Метрология, стандартизация и сертификация» для студентов очной формы обучения по направлениям: 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика», 09.03.04 «Программная инженерия», 10.03.01 «Информационная безопасность», может использоваться студентами, обучающимися по техническому направлению, профильному, «Стандартизация и метрология»

УДК 006.91(073)
ББК 30.10я73

ISBN 978-5-8088-1656-5

ВВЕДЕНИЕ

Развитие инструментов IT-индустрии позволяет значительно упрощать работу специалистов в различных сферах промышленности. Результатом внедрения специализированного программного обеспечения является автоматизация производств, внедрение интеллектуальных систем автоматизированной обработки данных и аналитики и др.

Пособие посвящено возможностям библиотек языка программирования *Python* для обработки экспериментальных результатов, построения графиков, диаграмм, проведения анализа данных, в том числе работе с нефизическими величинами, рассмотрены метрики качества программных решений, методов машинного обучения.

Первый раздел посвящен численным методам, позволяющим осуществлять обработку экспериментальных результатов: аппроксимировать экспериментальные точки, искать экстремумы функций, определять ошибки и выбросы.

Во втором разделе речь идет о методах машинного обучения, оценке их качества, обработке нефизических величин. Предложены примеры реализации программного кода для решения задачи предсказания ответа системы по заданным параметрам. Оценка качества такого предсказания является актуальной задачей современной цифровой метрологии.

В конце пособия приведены варианты для самостоятельного решения задач для закрепления пройденного материала. Работать над задачами рекомендуется на открытых ресурсах *GoogleColab* или *JupyterNotebook*. Рекомендуется также ознакомиться со следующими библиотеками языка программирования *Python: scikit-learn, matplotlib, seaborn, pandas, numpy*.

1. ЧИСЛЕННЫЕ МЕТОДЫ В МЕТРОЛОГИИ

1.1. Статическая обработка результатов прямых измерений многократными независимыми наблюдениями

Порядок и методику выполнения прямых измерений с многократными независимыми наблюдениями, обработки наблюдений и оценки их погрешностей регламентирует государственный стандарт [1].

Методы статистической обработки результатов измерений сводятся к определению числовых оценок параметров соответствующих законов распределения. Поэтому необходимо знание методов определения по экспериментальным данным числовых характеристик законов распределения. Рассеяние результатов при многократном измерении одной и той же величины постоянного размера является следствием множества причин, вклад каждой из которых незначителен по сравнению с суммарным действием всех остальных. Центральная предельная теорема теории вероятностей утверждает, что результат измерения при этом подчиняется нормальному закону. Это наиболее часто встречающееся распределение Гаусса [2]. Закон нормального распределения имеет фундаментальное значение для теории обработки результатов измерений.

К числу случайных величин, распределение которых подчиняется нормальному закону, относится большая часть случайных погрешностей. Существует две формы описания закона распределения случайной величины: дифференциальная и интегральная. Причем, в метрологии в основном используется дифференциальная форма закона распределения плотности вероятности случайной величины. Дифференциальная функция нормального закона распределения имеет вид:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-m)^2}{2\sigma^2}},$$

где $f(x)$ – плотность вероятности распределения x, $m=[M(x)]$ – математическое ожидание случайной величины x; σ – среднее квадратическое отклонение; $e=2{,}7183$ – основание натуральных логарифмов.

Интегральная функция нормального закона распределения

$$F(x)=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\chi} e^{-\frac{(x-m)^2}{2\sigma^2}}\,dx$$

В выражении для $f(x)$ входят две величины, значения которых полностью определяют закон распределения для каждого конкретного случая. Это m и σ.

При статистической обработке результатов наблюдений выполняют следующие операции:

1. Исключение известных систематических погрешностей из результатов наблюдений путем:

• ликвидации источников погрешностей до начала измерения;

• исключения погрешностей в процессе измерения способами замещения, компенсации погрешности по знаку, противопоставления, симметрических наблюдений;

• внесения вычисленных поправок в результаты измерения.

Прим. Результат наблюдений, в который введены поправки с целью устранения систематических погрешностей, считается исправленным.

2. Вычисление:

• среднего арифметического (центра распределения погрешностей) исправленных результатов наблюдений, принимаемого за результат измерения;

• оценки среднеквадратического отклонения результата наблюдения и измерения;

• доверительных границ случайной составляющей погрешности результата измерения (при этом проверяют гипотезу о том, что результат наблюдений принадлежит нормальному распределению).

Наиболее эффективной оценкой центра распределения погрешностей (математического ожидания) для распределения погрешностей, близких к нормальному закону, является среднее арифметическое x.

Несмещенной, состоятельной, эффективной оценкой для среднего m нормального распределения является выборочное среднее, определяемое по формуле

$$\overline{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$$

При значениях объема выборок $n \geq 20$ несмещенную оценку S для среднеквадратического отклонения σ определяют по формуле

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Оценку среднеквадратического отклонения результата измерений оценивают по формуле

$$S(\overline{x}) = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Отбраковка грубых и аномальных результатов проводится с целью исключения их из дальнейшей обработки. Если эти результаты не являются промахами, то необходимо подвергнуть результаты статистическому анализу.

Существуют различные критерии отбраковки. Наиболее часто употребляемый критерий основан на использовании значений интервала вероятности

$$\Phi(z) = \frac{1}{\sqrt{2\pi}}\int_0^z \exp\left(-\frac{t^2}{2}\right)dt$$

т. е. на предположении, что результаты измерений распределены по нормальному закону.

Сначала определяют среднее арифметическое значение и среднеквадратическое отклонение, для сомнительного результата x_C определяют величину

$$Z_c = \frac{|x_C - \overline{x}|}{S}$$

По таблице 1 находят значение $\Phi(z)$. Если значение $\Phi(z)$ близко к единицы, то результат считается грубым и его отбрасывают.

Частным случаем данного критерия является правило трех сигм. Погрешность $x_C - \overline{x}$ считается грубой, если она превосходит значения 3S.

Для определения эмпирического закона распределения от вариационного ряда необходимо перейти к статистическому или интервальному. Для этого вариационный ряд разбивают на N интервалов. Рекомендовано разбивать не более, чем 5 равных интервалов.

Таблица 1

Таблица значений интеграла вероятностей

Z_c	$\Phi(t)$	Z_c	$\Phi(t)$	Z_c	$\Phi(t)$
0,00	0,0000	1,25	0,7887	2,50	0,9876
0,05	0,0399	1,30	0,8064	2,55	0,9892
0,10	0,0797	1,35	0,8230	2,60	0,9907
0,15	0,1192	1,40	0,8385	2,65	0,9920
0,20	0,1585	1,45	0,8529	2,70	0,9931
0,25	0,1974	1,50	0,8664	2,75	0,9940
0,30	0,2358	1,55	0,8789	2,80	0,9949
0,35	0,2737	1,60	0,8904	2,85	0,9956
0,40	0,3108	1,65	0,9011	2,90	0,9963
0,45	0,3473	1,70	0,9109	2,95	0,9968
0,50	0,3829	1,75	0,9199	3,00	0,99730
0,55	0,4177	1,80	0,9281	3,10	0,99806
0,60	0,4515	1,85	0,9357	3,20	0,99863
0,65	0,4843	1,90	0,9426	3,30	0,99903
0,70	0,5161	1,95	0,9488	3,40	0,99933
0,75	0,5468	2,00	0,9545	3,50	0,99953
0,80	0,5763	2,05	0,9596	3,60	0,99968
0,85	0,6047	2,10	0,9643	3,70	0,99978
0,90	0,6319	2,15	0,9684	3,80	0,99986
0,95	0,6579	2,20	0,9722	3,90	0,99990
1,00	0,6827	2,25	0,9756	4,00	0,99994
1,05	0,7063	2,30	0,9786	4,10	0,99996
1,10	0,7287	2,35	0,9812	4,20	0,99997
1,15	0,7499	2,40	0,9836	4,40	0,99999
1,20	0,7699	2,45	0,9857	4,50	0,999994

При построении интервального ряда считают, что результаты, попавшие в интервал имеют одно и то же значение, соответствующее медианному значению. Для каждого интервала подсчитывает частотность.

Полученный результат оформляют графически в виде гистограммы, которая может быть построена с помощью библиотек *Python matplotlib*, функция *hist*(): https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.hist.html. По оси абсцисс откладываются интервалы, на них строятся прямоугольники высотой P_i

$$P_i = \frac{n_i}{n},$$

где n – число элементов в выборке, n_i – число элементов в интервале.

На диаграмме также часто обозначают среднее и медианное значение выборки.

Вследствие наглядности и информативности гистограммы часто применяют в производстве для оценки протекания технологических процессов. Гистограммы дают возможность зафиксировать состояние процесса в определенный момент времени. Полученная ступенчатая функция может быть аппроксимирована кривой с пиком или прямой. В первом случае речь идет о нормальном распределении, во втором – о равномерном.

Представим, что наша выборка есть ни что иное, как измерения случайных образцов деталей из партии. Ассиметрия гистограммы, сдвиг пика могут говорить о серьезных проблемах в производственном цикле.

Существует еще один метод оценки производственного процесса, который позволяет отслеживать состояние процесса во времени. Такой метод называется методом контрольных карт [3]. Контрольная карта представляет собой графическое изображение характеристики процесса, состоящее из центральной линии, контрольных границ и конкретных значений, имеющихся статистических данных, позволяющее оценить степень статической управляемости процесса.

Центральная линия – это среднее значение контролируемого параметра. Контрольные границы выбираются меньше значений поля допуска для того, чтобы провести корректирующие манипуляции над процессом до того, как значения выйдут за поле допуска. Как только значение выходит за контрольные границы считают, что процесс вышел из управляемого состояния. Пример контрольной карты Шухарта приведен на рис. 1. Иногда на контрольных картах обозначают поля допуска.

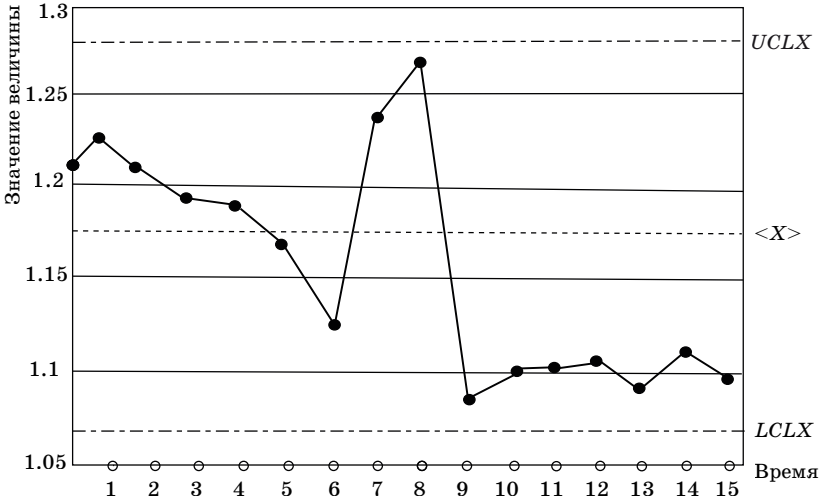

Рис. 1. Контрольная карта Шухарта: <X> – среднее значение, UCLX – верхняя контрольная граница, LCLX – нижняя контрольная граница

1.2. Аппроксимация экспериментальных точек функцией

Аппроксимация экспериментальных точек функцией является важным этапом при описании исследуемого процесса, позволяет находить грубые промахи, усреднять систематические погрешности, предсказывать физический или технологический процесс на основании экспериментальных результатов [4–5].

Пусть имеется некоторый набор точек x, каждой точке из набора соответствует некоторое значение $f(x)$. Такая функциональная зависимость может быть получена как экспериментально, так и путем выполнения некоторых дискретных расчетов. Например, при исследовании температуры воздуха в течение дня: температура фиксируется каждый час, то есть в каждый час мы имеем определённое значение функции температуры от времени.

Перед нами стоит следующая задача: заменить неизвестную, по сути дискретную, функцию $f(x)$ на непрерывной на некотором отрезке в любой точке функцию $F(x)$. Такое приближение называет аппроксимацией: замена одного дискретного объекта другим, непрерывным.

Существует два основных типа аппроксимации функции:

• интерполяция – функция $F(x)$ точно совпадает с дискретными значениями функции $f(x)$;

• метод наименьших квадратов – функция $F(x)$ может не совпадать с дискретными значениями функции $f(x)$, но быть максимально приближенной в среднем.

К методом интерполяции функции относятся линейная интерполяция, интерполяция полиномом, интерполяция сплайном.

В случае линейной интерполяции дискретные значения функции в смежных узловых точках соединяются отрезками прямых, фукнция $f(x)$ приближается ломаной с вершинами в данных точках. Получается, что каждый участок описывается своим особенным участком ломаной. График интерполирующей функции имеет изломы в узлах, поэтому получается достаточно высокая погрешность.

Решить эту проблему позволяет интерполяция полиномом Лагранжа [6]. Такая интерполяция дает высокую точность в случае, если значения функции в соседних узлах изменяются медленно. Интерполяционный полином имеет вид:

$$L_n(x) = \sum_{i=0}^{n} l_i(x)$$

Здесь $l_i(x)$ – полином степени n, составим следующим образом:

$$l_i(x) = C_i(x - x_0)\ldots(x - x_n)$$

Учитывая, что

$$C_i = \frac{y_i}{(x_i - x_0)\ldots(x_i - x_n)}$$

Получаем

$$l_i(x) = \frac{y_i(x - x_0)\ldots(x - x_n)}{(x_i - x_0)\ldots(x_i - x_n)} = y_i \prod_{k=0}^{n} \frac{x - x_k}{x_i - x_k}$$

Интеполяционный полимном Лагранжа имеет вид:

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{k=0}^{n} \frac{x - x_k}{x_i - x_k}$$

Выражение громоздкое, требует поточечной оценки, кроме того полиномиальная интерполяция не всегда удовлетворяет результатам. Увеличение интерполяционного многочлена не в каждом случае позволяет улучшить решение и уменьшить отклонение от истинного значения. Проблемой также может быть возникающая вертикальная асимптотика.

Во многих программных пакетах, позволяющих аппроксимировать экспериментальные точки, заложен алгоритм интерполяции сплайном. В таком способе используют гибкую кривую (еще ее называют линейкой, в переводе с английского), совмещенную с дискретными экспериментальными значениями.

Рассмотрим алгоритм на примере кубического сплайна, т. е. используется полином не выше третьей степени. Сплайн, совмещенный с узловыми значениями функции, проходит по линии, удовлетворяющей выражению

$$\phi^{(\mathrm{IV})}(x)=0$$

Функция $\phi(x)$ является полиномом, на каждом интервале имеет вид:

$$\phi_i(x)=a_i+b_i(x-x_{i-1})+c_i(x-x_{i-1})^2+d_i(x-x_{i-1})^3,$$

где a_i, b_i , c_i , d_i – коэффициенты сплайна, i – номер сплайна, по-другому называют также номер интервала.

Аппроксимируемая функция есть последовательность сплайнов, сшитых между собой на границах.

Рассмотрим метод наименьших квадратов. Представим аппроксимирующую функцию в виде:

$$F(x_i, a_0, a_1, \ldots, a_m)$$

Суть метода заключается в том, что необходимо получить такие значения нормирующих параметров $a_0, a_1, \ldots, a_m$, чтобы сумма квадратов отклоненений функции относительно дискретных значений была минимальна.

Рассмотрим алгоритм метода на примере полиномиальной аппроксимации. Аппроксимирующая функция есть полином вида:

$$F(x_i, a_0, a_1, \ldots, a_m)=a_0+a_1x+a_2x^2+\ldots+a_mx^m$$

Обозначим

$$\sum_{i=0}^{n}(y_i - F(x_i, a_0, a_1, \ldots, a_m))^2 = Q(a_0, a_1, a_2, \ldots, a_m)$$

Функция Q принимает минимальное значение, если частные производной этой функции по каждому параметру равны 0. Таким образом, для решения задачи необходимо решить следующую систему m уравнений (параметр jпринимает значения от 0 до m):

$$\sum_{i=0}^{n}(y_i - F(x_i, a_0, a_1, \ldots, a_m))^2 \frac{\partial}{\partial a_j} F(x_i, a_0, a_1, \ldots, a_m) = 0$$

Решением этой системы будут значения $a_0, a_1, \ldots, a_m$. Отсюда находится искомая аппроксимирующая функция $F(x_i, a_0, a_1, \ldots, a_m)$.

1.3. Определение локальных минимумов методом градиентного спуска

Определение локального минимума функции является важной задачей обработки экспериментального результата. Решение таких задач используется не только для оценки технологических процессов, но и в машинном обучении для нахождения минимального значения функции потерь. Функция потерь используется, чтобы контролировать ошибку в прогнозах модели машинного обучения. Поиск минимума означает получение наименьшей возможной ошибки или повышение точности модели [7].

Суть алгоритма заключается в осуществлении процесса получения наименьшего значения ошибки. Аналогично это можно рассматривать, как спуск с горы в самое низкое значение.

Предположим, имеется некоторая дифференцируемая функция с точкой экстремума (минимума). Справа от такой точки производная положительна, а слева – отрицательна. Предположим, что мы выбираем произвольную начальную точку на оси абсцисс. Двигаемся от этой точки к локальному минимуму. В этом случае в области положительных производных разницу между произвольной точкой и локальным минимумом будет уменьшаться, в области отрицательных производных – увеличиваться. Это можно описать следующим математическим выражением:

$$x_{n+1} = x_n - \frac{df(x)}{dx}, \quad n = 0, 1, 2, \ldots$$

Здесь n – номер итерации работы алгоритма.

Градиент уже учтен, когда определяли перемещение вдоль оси абсцисс для поиска оптимальной точки минимума. Но математика не терпит такой вульгарности, в ней все должно быть прописано и точно определено. Как раз для этого нужно брать не просто производную, а еще и определять направление движения, используя единичные векторы декартовой системы координат. В конечной формуле алгоритма поиска единичный вектор не пишется, вместо него указывается разность по оси ординат.

У метода есть один недостаток: значение производной может быть очень большим, в таком случае при реализации метода можно проскочить минимум, уйти дальше. Для решения этой проблемы вводят шаговое значение λ (шаг сходимости). Выражение принимает вид:

$$x_{n+1} = x_n - \lambda \frac{df(x)}{dx}, \quad n = 0,1,2,\ldots$$

В частном случае шаг сходимости может меняться в зависимости от итерации.

Для реализации алгоритма с помощью языка *Python* рекомендуется использовать библиотеки *numpy* и *matplotlib*.

2. МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ В МЕТРОЛОГИИ

2.1. Виды данных. Обработка данных. Анализ данных

Несколько десятков лет назад компьютеры резко подешевели и стали доступны для широкой аудитории, что произвело революцию как во многих отраслях науки, бизнеса и промышленности, так и в нашей повседневной жизни. С помощью компьютеров можно работать с огромными базами данных, автоматизировать бизнес-процессы, контролировать работу конвейера на производстве, упрощать управление самолетом или просто хранить коллекцию семейных фотографий.

За несколько десятков лет многие отрасли и компании накопили большие объемы данных, и теперь появилась возможность извлекать пользу из этих данных, находить в них нетривиальные закономерности. Методы машинного обучения и анализа данных все активнее используются при оптимизации производственных процессов и маршрутов транспорта, для оптимизации закупок и маркетинговых кампаний в интернет-коммерции, для создания новых лекарств и автомобилей с искусственным интеллектом.

2.1.1. Виды данных

Существует несколько основных типов данных, и важно знать, как можно работать с каждым из них, чтобы корректно осуществить сбор данных в форме, которая лучше всего удовлетворяет задаче. Существует много классификаций типов данных, но мы остановимся на таких уровнях измерения, как номинальный, порядковый, интервальный и нормативный. Разберем один простой пример.

Допустим, покупатель пришел в гипермаркет, ходит от отдела к отделу и кладет в корзину то, что ему необходимо: фрукты и овощи, мясо, рыбу, консервы, моющие средства и т. д. Покупатель может составить список, в котором было указано, из какого отдела он взял каждый продукт, такой список будет представлять собой данные номинального типа.

Номинальные данные можно посчитать, можно определить процент от целого, однако нельзя вычислить среднее значение. Сам термин «номинальный» имеет отношение к латинскому слову «*nomen*», которое означает «относящийся к именам». Мы называем этот вид данных номинальными, поскольку они содержат названия категорий, по которым распределяются данные. Номинальные

данные по определению не упорядочены; овощи как общая категория математически не больше и не меньше, чем молочная продукция. Мы можем сосчитать количество продуктов в корзине, взятых в молочном отделе, какой процент от покупок составляют овощи, но посчитать среднее значение каждого продовольственного отдела в корзине невозможно.

В случае, если доступны только две категории, данные относят к типу дихотомических. Ответы на вопросы, требующие ответа «да-нет», – это и есть дихотомические данные. Если, делая покупки, собираются данные о том, продавался товар со скидкой или нет, это и будут дихотомические данные.

В конце концов, покупатель приходит на кассу и хочет попасть в самую быструю очередь. Мысленно он разбивает очереди на короткие, средние и длинные. Поскольку такие данные естественным образом упорядочиваются по категориям, они называются порядковыми. Вопросы в анкетах, ответами на которые могут быть такие фразы, как «полностью не согласен», «не согласен», «нейтрально отношусь», «согласен», «полностью согласен», предназначены для сбора порядковых данных. Ни одна из категорий порядковой шкалы не имеет фактической математической величины. Числовые значения зачастую присваиваются категориям для того, чтобы облегчить запись или анализ данных (например: 1 = полностью не согласен, 5 = полностью согласен), но это распределение условно, и вы можете выбрать любую группу упорядоченных чисел для обозначения групп. Например, вы с такой же легкостью можете решить, что цифра 5 будет обозначать «полностью не согласен», а 1 – «полностью согласен». Цифры, которые вы присваиваете порядковым категориям, влияют на толкование конечного анализа, но вы можете выбрать любой набор цифр, при условии соблюдения порядка нумерации.

Так же как и номинальные данные, порядковые данные можно посчитать и определить процент от целого, однако нет единого мнения о том, можно ли для порядковых данных посчитать среднее значение. С одной стороны, невозможно определить среднее значение для категории «полностью согласен», например, и даже если вы определите их числовые значения, они не будут иметь фактической математической величины. Каждое числовое значение представляет определенную категорию, а не количество чего бы то ни было. С другой стороны, если принять, что разница величин между последовательными категориями приблизительно одинаковая (например, разница между «полностью не согласен» и «не согласен»

такая же, как и между «не согласен» и «отношусь нейтрально», и так далее), и для обозначения категорий используются последовательные числа, тогда среднее значение ответов тоже можно интерпретировать применительно к той же шкале.

Вернемся в магазин. Покупатель стоит в очереди достаточно долго и смотрит на часы, чтобы узнать, сколько именно. Он стал в очередь в 11:15, а сейчас 11:30. Время суток – считается интервальными данными. Этот вид данных называется так, потому что интервалы между точками измерения одинаковы. Поскольку в каждой минуте 60 секунд, разница между 11:15 и 11:30 такая же, как между 12:00 и 12:15. Интервальные данные – числовые, поэтому вы можете производить с ними математические операции, однако такие данные не имеют «значимой» нулевой точки – то есть при значении ноль то, что вы измеряете, не отсутствует. 0:00 часов означает не отсутствие времени, а начало нового дня. Другие интервальные данные, с которыми вы сталкиваетесь в повседневной жизни, это календарный год и температура. Нулевое значение для годов не значит, что ранее времени не существовало, а нулевая температура (измеряемая в градусах Цельсия или Фаренгейта) отнюдь не показатель того, что тепла нет.

Увидев, что на часах 11:30, покупатель подумал: «Неужели я стою в очереди уже 15 минут?» Когда мы говорим о времени в таком контексте, это уже нормативные данные. Нормативные данные – числовые, и имеют много общего с интервальными данными, кроме того, что нормативные, в отличие от интервальных, имеют значимую нулевую точку. В нормативных данных ноль означает отсутствие того, что вы измеряете, – ноль минут, ноль людей в очереди, ноль молочных продуктов в вашей корзине. Во всех этих случаях ноль означает, что у вас нет того, что вы измеряете, и это отличается от того, что мы обсуждали в разделе интервальных данных. Другие часто встречаемые переменные, которые можно отнести к нормативным данным, – рост, вес, возраст и деньги.

Интервальные и нормативные данные могут быть либо дискретными, либо непрерывными. Дискретные данные выражены ограниченным набором значений (обычно целыми числами), величины между этими значениями невозможны. В очереди должно быть целое число людей, в ней не может быть одной трети человека. У вас может получиться в среднем по 4,25 человека в каждой очереди, но фактическое количество людей должно быть целым числом. Непрерывные данные могут принимать любое значение на шкале. Вы можете купить 1,25 фунта сыра или стоять в очереди 7,75 минут.

Это не значит, что данные могут принимать все возможные числовые значения – только все значения в рамках границ шкалы. Вы не можете стоять в очереди отрицательный промежуток времени и не можете купить отрицательное количество унций сыра, но тем не менее, эти данные – непрерывны.

Часто переменные описывают, как один из вышеперечисленных видов данных. Многие переменные не относятся к какому-либо определенному виду данных. Чаще всего вид данных определяется методом их сбора.

Давайте рассмотрим переменную возраста. Данные о возрасте обычно собирают как нормативные, однако их также можно собрать и как порядковые. Это происходит, когда в анкетах спрашивают: «К какой возрастной группе вы относитесь?» В таком опросе у вас не будет данных о возрасте каждого отдельного респондента, вы только сможете узнать, скольким из них было между 18–24 годами, 25–34 и так далее. Вы можете собрать показатели холестерина респондентов для медицинского исследования, либо просто спросить участников опроса, повышен у них холестерин или нет. То есть, это одна переменная и два разных метода сбора данных – и два различных вида данных.

Общее правило состоит в том, что вы можете двигаться вниз по уровню измерения, но не вверх. Если можно собирать переменные как интервальные или рациональные данные, их также можно собирать как номинальные или порядковые данные, но, если переменная номинальная по своей природе, как отдел в супермаркете, вы не можете собирать ее как интервальные, порядковые или нормативные данные. Переменные, имеющие порядковую природу, можно собирать как номинальные данные, но не как интервальные или нормативные. Однако, многие переменные, собираемые как порядковые данные, имеют схожую переменную, которую при желании можно собирать как интервальные или нормативные данные.

Важно помнить, что общее правило «двигаться можно вниз, но не вверх», применимо и во время анализа и визуализации данных. Если вы собираете переменную как нормативные данные, вы всегда можете позже сгруппировать данные для визуализации, если этого требует ваша работа. Если же вы собираете ее на более низком уровне измерения, позже вы не сможете перейти на более высокий уровень, не собрав больше данных. Например, если вы решили собирать данные о возрасте как порядковые данные, вы не сможете позже посчитать средний возраст, и ваша визуализация будет огра-

ничена демонстрацией возрастных групп; вы не сможете показать возраст как непрерывные данные.

Существуют еще термины, часто используемые применимо к видам данных. Ранее мы говорили о номинальных и порядковых данных как о способе распределить данные по категориям. Некоторые источники считают, что оба типа принадлежат к категориальным данным, где номинальные данные – неупорядоченные категориальные данные, а порядковые – упорядоченные. Другие источники относят к категориальным данным только номинальные, и считают, что понятия «номинальные данные» и «категориальные данные» – взаимозаменяемы. Эти источники относят порядковые данные к отдельной группе.

Качественные данные относятся к нечисловым данным, в то время как количественные данные – числовые и, соответственно, поддающиеся счету. Качественные данные всегда требуют предобработки или других методов анализа, в отличие от количественныхданных. Примерами могут считаться записи прямого наблюдения либо транскрипты интервью. Подобным образом, интервальные и нормативные данные всегда считаются количественными, поскольку они всегда числовые. Однако есть некое расхождение во мнениях относительно номинальных и порядковых типов данных. Некоторые источники называют их качественными, так как их категории описательные, а не числовые. Однако, поскольку эти данные можно посчитать и использовать для подсчета процентов, другие источники считают их количественными, поскольку они в этом смысле поддаются счету.

2.1.2. Анализ данных

Анализ данных и их предварительная подготовка перед обработкой для получения некоторой модели, оценка вероятности правильного решения задачи на основе имеющихся данных является одной из первостепенных задач машинного обучения.

Анализ данных – это процесс сбора, преобразования, очистки и моделирования данных с целью получения необходимой достоверной информации. Часто используют визуализацию данных для их наглядного представления в виде рисунков, схем. Термины «Моделирование данных» и «Анализ данных» означают одно и то же. Процесс анализа данных состоит из следующих этапов, которые тесно связаны друг с другом по своей природе:

- Спецификация требований к данным
- Сбор информации

- Обработка данных
- Очистка данных
- Анализ данных
- Обсуждение результатов

Спецификация требований к данным: данные, необходимые для анализа, основаны на опросе или эксперименте. В зависимости от поставленной задачи выбираются входные данные для анализа, например, набор данных (датасет) «Население земли». В датасете указаны конкретные переменные, относящиеся к населению, например, возраст, пол, раса и доход. Данные могут быть числовыми или категориальными.

Сбор данных – это процесс сбора информации по целевым переменным по определенным требованиям. Акцент делается на обеспечении точного и честного сбора данных. Сбор данных обеспечивает как основу для измерения, так возможность получения более точных результатов. Данные могут собираться из различных источников: от баз данных организаций до информации от интернет-ресурсов. Полученные таким образом данные могут не быть структурированными и содержать нерелевантную информацию. Следовательно, собранные данные необходимо подвергнуть обработке и очистке.

Обработка данных включает в себя структурирование данных в соответствии с требованиями соответствующих инструментов анализа. Например, данные могут быть размещены в строках и столбцах таблицы Excel или статистическом приложении. Возможно, потребуется создать модель данных.

Обработанные и упорядоченные данные могут быть неполными, содержать дубликаты или ошибки. Очистка данных – это процесс предотвращения и исправления ранее перечисленных недостатков. Существует несколько способов очистки данных в зависимости от вида данных. Например, имеется база стоимости квартир, продавец на сайте ошибся на порядок и выставил квартиру за достаточно высокую сумму. В таком случае мы можем выкинуть данные. Недостающие данные могут быть доопределены путем заполнения средним значением, указанным достоверными источниками (кадастровая стоимость квартиры), или известными пороговыми значениями.

Данные, которые обрабатываются, упорядочиваются и очищаются, будут готовы для анализа. Результаты анализа данных должны быть представлены в формате, необходимом пользователям для принятия решений и дальнейших действий. Отзывы пользо-

вателей могут привести к дополнительному анализу. Аналитики данных могут выбирать методы визуализации данных, такие как таблицы и диаграммы, которые помогают ясно и эффективно донести сообщение до пользователей. Инструменты анализа позволяют выделить необходимую информацию с помощью цветовых кодов и форматирования в таблицах и диаграммах.

В ходе обработки данных можно пользоваться следующим алгоритмом [8–9].

1. Загрузить датасет, используя библиотеку *pandas*.

```
#датасет из файла
import pandas as pd
data = pd.read_csv ('my file.csv')
```

Вывести на экран данные `data.head()`.

2. Получить информацию о данных `data.info()`. Проанализировать данные, например, на наличие пустых ячеек.

3. Осуществить анализ данных в соответствии с задачами. Необходимо выбрать наиболее важные данные, объединить похожие, реструктуризировать имеющиеся, привести текстовые данные в численный вид, убрать несущественные данные. Заполнить пропуски (это можно сделать либо с помощью среднего или медианного значения, либо 0, либо просто выкинуть строки, если их мало).

4. Используя библиотеку *matplotlib* и *seaborn* визуализировать данные в виде диаграмм, схем, гистограмм и графиков. Аргументировать выбор основных признаков для выполнения будущего предсказания. Использовать можно гистограммы, диаграммы, корреляция признаков наглядно представляется с помощью тепловой карты.

2.2. Методы машинного обучения и оценка качества моделей

Методы машинного обучения применяются в исследованиях и отраслях промышленности для составления прогнозов. Эта область постоянно развивается: появляются и разрабатываются новые методологии. Остановимся на классических и самых простых. Простота методов не говорит об их неэффективности, зачастую даже на первый взгляд сложные задачи можно решать классическими методами.

Большинство классических алгоритмов уже прописаны в *Python* в библиотеке *sklearn* (https://scikit-learn.org/stable/), что значительно упрощает решение задач.

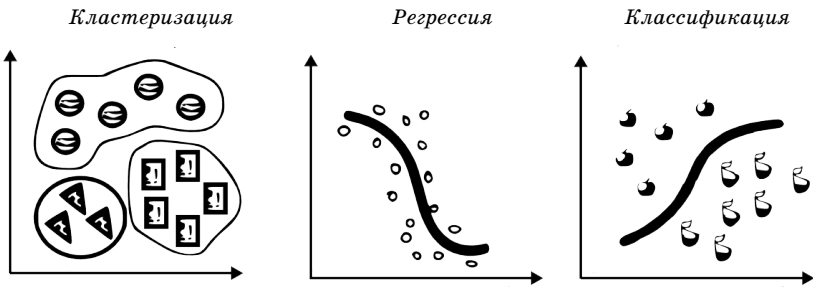

Рис. 2. Методы машинного обучения

Алгоритм машинного обучения, также называемый моделью, представляет собой математическое выражение. Цель – перейти от данных к анализу. Например, если интернет-магазин хочет прогнозировать продажи на следующий квартал, он может использовать алгоритм машинного обучения, который прогнозирует эти продажи на основе прошлых продаж и других соответствующих данных. Точно так же банковский специалист в сфере безопасности на основе данных о заемщике может дать резюме касательно выдачи или отказе в кредитном займе. С помощью видеоизображений зданий и модели инженер анализирует состояние строения и во время производит работы.

К основным методам машинного обучения относятся регрессия, классификация и кластеризация (рис. 2).

2.2.1. Регрессия (Regression)

Методы регрессии относятся к категории контролируемого машинного обучения (то есть мы заранее имеем достоверные данные, на которых обучается модель). Они помогают предсказать или объяснить конкретное числовое значение на основе набора предшествующих данных, например, прогнозирование цены собственности на основе предыдущих данных о ценах на аналогичные объекты.

Самый простой метод – это линейная регрессия (*LinearRegression*), при которой мы используем математическое уравнение линии ($y = m \cdot x + b$) для моделирования набора данных. Мы обучаем модель линейной регрессии с множеством пар данных (X, y), вычисляя положение и наклон линии, которые минимизируют общее расстояние между всеми точками данных и линией. Другими словами, мы вычисляем наклон (m) и точку пересечения по оси y(b) для линии, которая наилучшим образом аппроксимирует наблюде-

ния в данных. В библиотеке *sklearn* приведены различные вариации линейной регрессии https://scikit-learn.org/stable/modules/linear_model.html .

Пример вызова линейной модели:

Достаточно хорошо работают методы *SVD, Lasso*, стохастический градиентный спуск, деревья решений, но на их обучение может потребоваться некоторое существенное время при больших объемах данных. Ознакомиться с ними можно по ссылке https://scikit-learn.org/stable/modules/linear_model.html#.

2.2.2. Классификация

Другой класс контролируемого машинного обучения – это методы классификации. Они предсказывают или определяют принадлежность к классу. Например, они могут помочь предсказать, купит ли продукт покупатель в интернет-магазине. Выход может быть да или нет: купит или не купит.

Но методы классификации не ограничиваются двумя классами. Например, метод классификации может помочь оценить, содержит ли данное изображение автомобиль или грузовик. В этом случае на выходе будут 3 разных значения:

1) изображение содержит автомобиль;

2) изображение содержит грузовик;

3) или изображение не содержит ни автомобиля, ни грузовика.

Самый простой алгоритм классификации – это логистическая регрессия, что звучит как метод регрессии, да и в библиотеке *sklearn*, правда с комментарии, тоже значится в разделе «Линейные модели» (*Linear Models*), но это не так. Логистическая регрессия оценивает вероятность возникновения события на основе одного или нескольких входных данных.

Логистическая регрессия может принимать в качестве входных данных два экзаменационных балла студента (математика и физика), чтобы оценить вероятность того, что студент будет принят в конкретный ВУЗ. Поскольку оценка является вероятностью, на выходе получается число от 0 до 1, где 1 представляет полную достоверность. Для студента, если оценочная вероятность больше 0,5, то мы прогнозируем, что он или она будут приняты. Если предполагаемая вероятность меньше 0,5, мы прогнозируем, что ему или ей будет отказано.

На рис. 3 показаны оценки студентов за прошлые года, а также то, были ли они приняты. Логистическая регрессия позволяет нам провести линию, которая представляет границу принятия решения: поступили или нет.

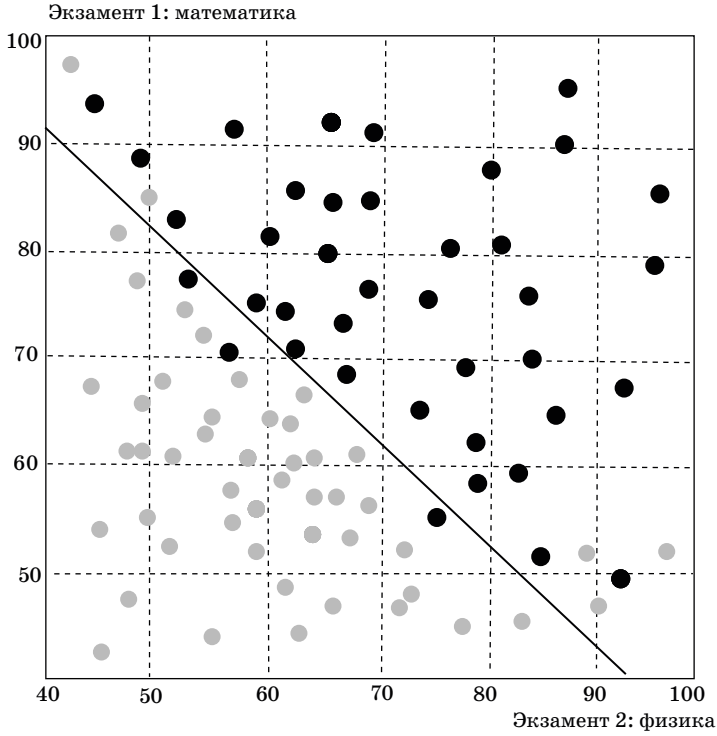

Рис. 3. Иллюстрация к задаче классификации

Логистическая регрессия – это простейшая модель классификации.

Пример вызова логистической регрессии

```
from sklearn.linear_model import LogisticRegression
LogR = LogisticRegression (random_state = 0).fit(X,y)
```

Также при решении задачи можно попробовать нелинейные классификаторы, такие как деревья решений, случайные леса и др. Деревья решений (`from sklearn.tree import DecisionTreeClassifier`) – это непараметрический контролируемый метод обучения, используемый для классификации и регрессии. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных.

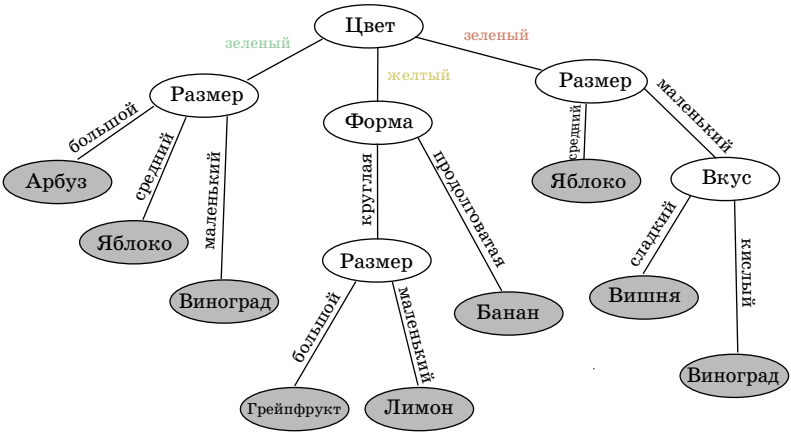

Рис. 4. Дерево решений

Например, деревья решений обучаются на основе данных, чтобы аппроксимировать синусоидальную кривую с помощью набора правил принятия решений «если-то-еще». Чем глубже дерево, тем сложнее правила принятия решений и тем лучше модель.

2.2.3. Кластеризация

С помощью методов кластеризации мы можем обучаться без предварительной информации о процессе, потому что цель таких алгоритмов – сгруппировать или объединить наблюдения со схожими характеристиками. Методы кластеризации не используют выходную информацию для обучения, а вместо этого позволяют алгоритму определять выходные данные. В методах кластеризации мы можем использовать визуализации только для проверки качества решения.

Наиболее популярным методом кластеризации является метод средних *K-mean*, где «*K*» представляет количество кластеров, которые пользователь выбирает для создания.

2.2.4. Оценка качества модели

Метрики качества для задачи класификации

В задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются метрики, а их выбор и анализ – непременная часть работы. Рассмотрим некоторые критерии качества в задачах классификации, обсудим, что является важным при выборе метрики и что может пойти не так.

В логистической регрессии порог отсечения изменяется от 0 до 1 – это и есть расчетное значение уравнения регрессии. Будем называть его рейтингом. Для понимания сути ошибок I и II рода рассмотрим четырехпольную таблицу сопряженности (*confusion matrix*), которая строится на основе результатов классификации моделью и фактической (объективной) принадлежностью примеров к классам.

Таблица 2

Матрица сопряженности

	Корректно (соответствует 1 в обучающей выборке) Actual positive	Некорректно (соответствует 0 в обучающей выборке) Actual negative
Корректное предсказание (*predicted positive*)	*TP*	*FP*
Некорректное предсказание (*predicted negative*)	*FN*	*TN*

• *TP* (*True Positives*) – верно классифицированные положительные примеры (так называемые истинно положительные случаи).

• *TN* (*True Negatives*) – верно классифицированные отрицательные примеры (истинно отрицательные случаи).

• *FN* (*FalseNegatives*) – положительные примеры, классифицированные как отрицательные (ошибка I рода). Это так называемый «ложный пропуск» – когда интересующее нас событие ошибочно не обнаруживается (ложно отрицательные примеры).

• *FP* (*FalsePositives*) – отрицательные примеры, классифицированные как положительные (ошибка II рода). Это ложное обнаружение, т. к. при отсутствии события ошибочно выносится решение о его присутствии (ложно положительные случаи).

Матрица может быть вызвана из библиотеке *sklearn* следующей функцией

```
import sklearn.metrics
#эталонныеметки
y_true = [“positive”,”negative”,”negative”,”positive”,”posi
tive”,”positive”,”negative”]
# предсказанныеметки
y_pred = [“positive”,”negative”,”positive”,”positive”,”nega
tive”,”positive”,”positive”]
r = sklearn.metrics.confusion_matrix(y_true, y_pred)
print(r)
```

Количество положительных ответов можно определить

$$P = TP+FN$$

Количество отрицательных ответов

$$N = TN+FP$$

Существуют следующие метрики:

1. *Accuracy* – доля правильно предсказанных от всех вариантов

$$Accuracy = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+FP+FN+TN}$$

2. *Precision* – доля правильно предсказанных среди причисленных моделью к категории 1

$$Precision = \frac{TP}{TP+FP}$$

3. *Recall* – доля правильно предсказанных среди категории 1

$$Recall = \frac{TP}{TP+FN}$$

Метрика *Accuracy* вызывается следующим образом `LogR.score(X_test, Y_test)`.

Часто для задач классификаций используют *ROC* – кривую (кривая рабочих характеристик, *Receiver Operating Characteristics curve*) для анализа поведения классификаторов при различных пороговых значениях. Показывает долю ложно положительных примеров в сравнении с долей истинно положительных примеров. Название пришло из систем обработки сигналов. Поскольку классов два, один из них называется классом с положительными исходами, второй — с отрицательными исходами. *ROC*-кривая показывает зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров. В терминологии *ROC*-анализа первые называются истинно положительным, вторые – ложно отрицательным множеством. При этом предполагается, что у классификатора имеется некоторый параметр, варьируя который, мы будем получать то или иное разбиение на два класса. Этот параметр часто называ-

ют порогом, или точкой отсечения (*cut-off value*). В зависимости от него будут получаться различные величины ошибок первого или второго рода.

Для реализации *ROC*-анализа вычисляют:

1) *True Positive Rate* (*TPR*) – доля истинно положительных примеров.

$$TRP = Recall = \frac{TP}{TP + FN}$$

2) *False Positive Rate* (*FPR*) – доля ложно положительных примеров. В идеале корреляция *TPR* и *FPR* выглядит следующим образом:

$$FRP = \frac{FP}{FP + TN}$$

Идеальный случай приведен на рис. 5.

ROC-анализ тесно связан с бинарной логистической регрессией и применяется для оценки качества моделей. Он позволяет выбрать модель с наилучшей прогностической возможностью, проанализировать чувствительность и специфичность моделей, подобрать порог отсечения.

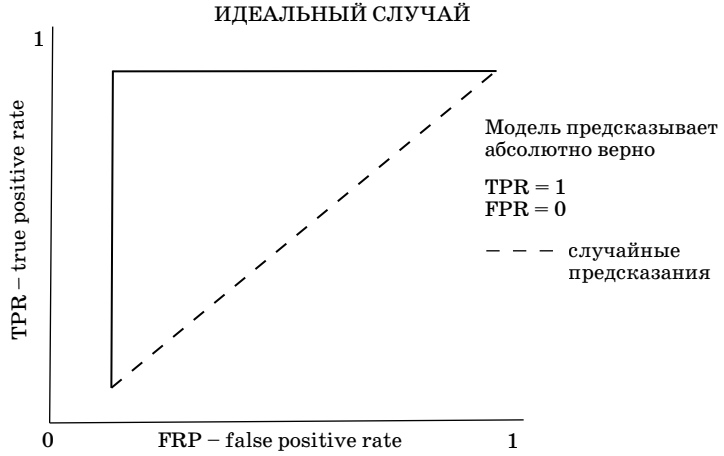

Рис. 5. Идеальная кривая ROC

В *Python* заложены алгоритмы:

```
from sklearn.metricsimportroc_curve
from sklearn.metrics import roc_auc_score
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
LogR = LogisticRegression (random_state = 0).fit(X,y)
```

Для каждой обученной модели классификации можно определить *TPR*, *FPR* по правилу:

```
fpr, tpr, thres = roc_curve (реальное значение Y, прогноза
модели LR_pred[:, 1] )
roc_auc_score( Y_test, LR_pred[:, 1] )
#выведетзначениеошибки
```

Пример кривой *ROC* для различных моделей-классификаторов приведен ниже на рис. 6.

Метрика силуэта хорошо работает для кластеризации. Значение силуэта показывает, насколько объект похож на свой кластер по сравнению с другими кластерами.

```
from sklearn.metrics import silhouette_score silhouette_
score(X_test, KM_pred).
```

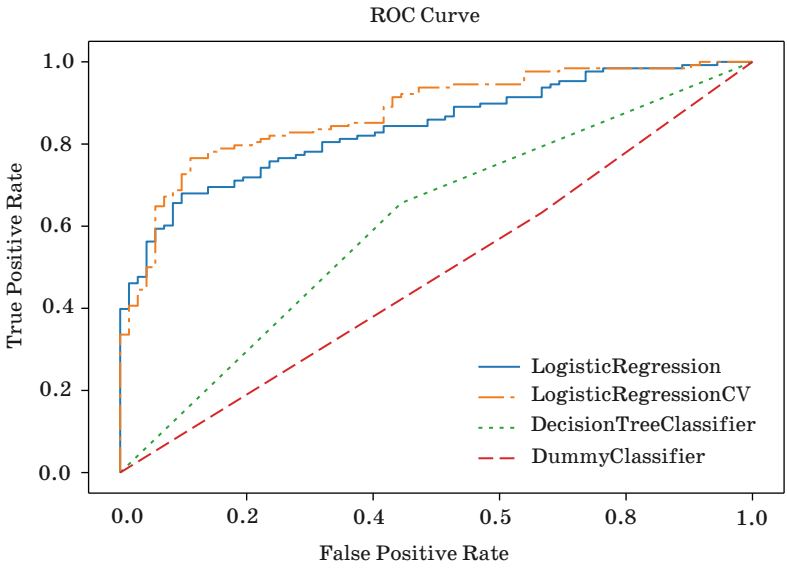

Рис. 6. Пример кривой ROC для решения одной задачи разными методами классификации

С работой различных метрик можно подробнее ознакомиться в библиотеке *sklearn*: https://scikit-learn.org/stable/modules/model_evaluation.html.

Метрики качества для задачи кластеризации и логистической регрессии

Для оценки регрессии наиболее часто используемыми мерами качества являются

- средняя квадратичная ошибка (*Mean Squared Error, MSE*)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(a(x_i) - y_i\right)^2$$

- средняя абсолютная ошибка (*Mean Absolute Error, MAE*)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|a(x_i) - y_i\right|$$

- коэффициент детерминации (определяет долю дисперсии, нормированная среднеквадратическая ошибка)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(a(x_i) - y_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2}$$

- средняя абсолютная процентная ошибка (*Mean Absolute Percentage Error, MAPE*), проблема ошибки – нестабильность

$$MAPE = 100\% \cdot \frac{1}{n}\frac{\sum_{i=1}^{n}\left|a(x_i) - y_i\right|}{\sum_{i=1}^{n}\left|y_i\right|}$$

- корень из средней квадратичной ошибки (*Root Mean Squared Error, RMSE*)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2}$$

Хороший способ оценки модели предусматривает применение кросс-валидации (скользящего контроля или перекрестной проверки). В таком случае обучающая выборка делится на две подвыборки: обучающую и контрольную. Разбиение делается несколько раз, для каждого разбиения выводится результат оценки. Пример применеия кросс-валидации:

```
from sklearn import datasets, linear_model
fromsklearn.model_selection import cross_val_score
diabetes = datasets.load_diabetes()
X = diabetes.data[:150]
y = diabetes.target[:150]
lasso = linear_model.Lasso()
#выведим значение для трех скользящих разбиений
print(cross_val_score(lasso, X, y, cv = 3))
```

2.3. Agile и отчетность

Agile – это итеративный подход к управлению проектами и разработке программного обеспечения, который помогает командам предоставлять своим клиентам все самое важное в сжатые сроки и по первому требованию. Принцип работы основан не на быстром решении итоговой задачи, а на гибком командном подходе, пошаговой работе. Каждый шаг, требования к каждой подзадаче, планы и результаты постоянно оцениваются как заказчиком, так и членами команды. Это позволяет быстро реагировать на любые запросы и вносить корректировки. *Agile* требует совместной работы команды: открытое общение, сотрудничество, адаптация и доверие между членами команды.

Agile – это группа методологий, которые демонстрируют приверженность жестким циклам обратной связи и постоянному совершенствованию.

Agile Manifesto, в котором изложены основные идеи, не предписывает четких сроков выполнения подзадач или количество участников в проекте.В нем изложены основные ценности, которые ставят людей на первое место.

Методология *Agile* позволяет обеспечивать не только постоянную обратную связь с заказчиком и внутри команды, но и правильное формирование документооборота. Проекты могут храниться на *github* или на облачных хранилищах. Документы проекта имеют четкую структуру. Каждый файл должен иметь четкое название, автора. Все рабочие документы не удаляются, а остаются на хранении. Файлы с данными, кодом, презентациями хранятся в разных каталогах (папках). К каждому укрупненному каталогу обязательно прилагается файл *readme.txt* для быстрого ознакомления с содержимым. Пример содержания корневого каталога на *github*:

1. *readme.txt* (указывается содержание каталога, комментарии).
2. *code* (каталог с кодом).

3. *data* (данные).

4. *report* (каталог, в котором хранятся презентации, отчеты заказчику, презентации коллегам).

Отчеты имеют строгую структуру следующего содержания вне зависимости от формата оформления: будь то презентация или текстовый документ:

1. Постановка задачи, автор.

2. Анализ данных, предобработка (обычно добавляют графики, рисунки).

3. Модель обучения и результат, оценка качества модели.

4. Улучшения качества модели, рекомендации.

2.4. Пример оценки качества модели линейной регрессии

Рассмотрим на примере задачи, в которой требуется предсказать оценку вина на основании физико-химических свойств (https://www.kaggle.com/rajyellow46/wine-quality).

1. Импорт необходимых библиотек

```
# Ignore warnings
Import warnings
warnings.filter warnings('ignore')
# Handle table-like data and matrices
Import numpy as np
import pandas as pd
# Modelling Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
# Modelling Helpers
from sklearn.preprocessing import Normalizer,scale
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Visualisation
Import matplotlib.pyplot as plt
Import seaborn as sns
```

2. Вспомогательные функции для визуализации данных

```
defplot_correlation_map(df):
corr = wine.corr()
_,ax = plt.subplots(figsize = (12,10))
cmap = sns.diverging_palette(220,10,as_cmap = True)
_ = sns.heatmap(
corr,
cmap = cmap,
square = True,
cbar_kws = {'shrink':.9},
ax = ax,
annot = True,
annot_kws = {'fontsize':12}
)

defplot_variable_importance(X,y):
tree = DecisionTreeClassifier(random_state = 99)
tree.fit(X,y)
plot_model_var_imp(tree,X,y)
defplot_model_var_imp(model,X,y):
imp = pd.DataFrame(
model.feature_importances_,
columns = ['Importance'],
index = X.columns
)
imp = imp.sort_values(['Importance'],ascending = True)
imp[:10].plot(kind = 'barh')
print(model.score(X,y))
```

3. Загружаем данные из файла

```
wine = pd.read_csv("winequalityN.csv")
wine.shape
```

4. Статистика и визуализации

```
wine.head()# вывод таблицы
```

Получим информацию о незаполненных ячейках

```
wine.info()
out:
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):

 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   type                  6497 non-null   object
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
wine.describe()
```

Рассмотрим распределения признаков и оценок вина в виде диаграмм:

```
cols = ['fixed acidity','volatile acidity','citric acid','residual
sugar','chlorides','free sulfur dioxide',
'total sulfur dioxide','density','pH','sulphates','alcohol']
For feature in cols:
wine[feature].hist(bins = 25)
plt.xlabel(feature)
plt.ylabel("Count")
plt.title(feature)
plt.show()
```

```
sns.countplot(wine['quality'])
```

Тип вина задан в виде категориальной переменной, и выше не рассмотрено влияние этого параметра, посмотрим, как разделены белое и красное вино

```
sns.countplot(wine['type'])
```

Белое вино сильно популярнее красного. Воспользуемся *Label Encoder* для перевода из категориальных переменных в числовые и построим тепловую карту для оценки корреляции признаков

```
#Data transforamation
label = LabelEncoder()
dicts = {}

label.fit(wine.type.drop_duplicates())
dicts['type'] = list(label.classes_)
wine.type = label.transform(wine.type)

plot_correlation_map(wine)
```

С оценкой вина очень связаны признаки *alcohol*, *density*, *chlorides*, *volatileacidity*, *type*. Поскольку люди предпочитают белое вино, и переменная сильно не влияет на категорию, то *type* не учитываем.

```
wine.pivot_table('quality','type').plot(kind = 'bar',stacke
d = True)
```

5. Подготовка данных. Заполняем пропущенные значения в переменных, только «важных» из списка признаков *alcohol*, *density*, *chlorides*, *volatileacidity*, *type* средним значением

```
wine.isnull().sum()
type                   0
fixed acidity         10
volatile acidity       8
citric acid            3
residual sugar         2
chlorides              2
free sulfur dioxide    0
total sulfur dioxide   0
density                0
pH                     9
sulphates              4
alcohol                0
quality                0
dtype: int64

# Fill missing values
wine['chlorides'] = wine.chlorides.fillna(wine.chlorides.
mean())
```

```
wine = wine.rename(columns = {'volatile acidity':'volatile_
acidity})
wine['volatile_acidity'] = wine.chlorides.fillna(wine.
volatile_acidity.mean())
wine.head()
wine.isnull().sum()
```

6. Подготовка финального датасета для моделирования.

Выбираем признаки для формирования итогового датасет. У нас доступны признаки:

- volatile acidity;
- alcohol;
- density;
- chlorides.

```
# features for a model
#features = ['type', 'alcohol', 'density', 'chlorides']
features = ['alcohol','density','chlorides','volatile_
acidity','type']
X = pd.get_dummies(wine[features])
# target
y = wine['quality']
train_X,valid_X,train_y,valid_y = train_test_
split(X,y,train_size = 0.8)
plot_variable_importance(train_X,train_y)
features = ['alcohol','density','chlorides','volatile_
acidity']
X = pd.get_dummies(wine[features])
```

Разделяем выборку на тестовую и обучающую

```
train_X,valid_X,train_y,valid_y = train_test_
split(X,y,train_size = 0.8)
```

7. Моделирование. Теперь выберем модель, которую хотели бы обучить. Используем обучающий набор данных для модели и затем проверим ее с помощью тестового набора. Остановимся на линейной модели: *LR* – линейная регрессия

```
LR = LinearRegression()
X, y = make_regression(n_features = 4, n_
informative = 2,random_state = 0,shuffle = False)
regr = RandomForestRegressor(max_depth = 2,random_
state = 0)
```

Обучение моделей

```
LR.fit(train_X,train_y)

#Оценим решение на тестовой выборке
print(«Модели регрессии»)
print(«Линейнаярегрессия:»,»наобучающейвыборке    -    «,LR.
score(train_X,train_y),»,    натестовойвыборке    -    «,LR.
score(valid_X,valid_y))

out:
Модели регрессии
Линейная регрессия: на обучающей выборке - 0.21, на тестовой
выборке - 0.18
```

Результат неудовлетворительный, необходимо настроить линейную модель или попробовать использовать другие модели.

3. ЗАДАНИЯ ДЛЯ ЗАКРЕПЛЕНИЯ НАВЫКОВ

Задание № 1. К разделу 1.1.

Имеются некоторые числовые результаты измерений деталей на производстве. Необходимо определить грубые промахи, построить гистограмму и определить тип распределения. Построить контрольную карту Шухарта. Сделать вывод о технологическом процессе.

Вариант № 1

0, 9879, 0.9823, 0.9962, 0.9736, 1.0000, 1.0039, 1.0080, 0.9800, 0.9923, 0.9952, 1.0098, 1.0155, 0.9969, 0.9995, 1.0143, 1.0113, 1.0035, 0.9933, 0.9873, 1.0130, 1.0068, 1.0047, 1.0011, 0.9981, 1.0015, 0.9989, 0.9856, 1.0026, 0.9889, 1.0061, 0.9903, 1.0071, 1.0085, 1.0195, 1.0088.

Вариант № 2

0.9984, 1.0074, 1.0143, 0.9881, 0.9740, 0.9964, 1.0131, 1.0029, 0.9972, 1.0118, 0.9911, 1.0203, 0.9955, 0.9808, 1.0094, 0.9827, 1.0063, 0.9993, 1.0039, 0.9879, 0.9900, 1.0086, 0.9938, 1.0051, 0.9998, 1.0014, 1.0005, 0.9857, 1.0170, 1.0021, 1.0035, 1.0103, 1.0075, 1.0031, 1.0007.

Вариант № 3

1.0069, 1.0081, 0.9924, 1.0173, 0.9958, 0.9840, 0.9929, 1.0083, 1.0068, 0.9886, 0.9775, 1.0128, 1.0107, 1.0025, 0.9862, 1.0038, 0.9998, 0.9978, 0.9968, 1.0014, 0.9984, 0.9994, 1.0262, 1.0069, 1.0053, 1.0046, 0.9809, 0.9879, 1.0150, 0.9901, 1.0143, 0.9948, 0.9917, 1.0096, 1.0087.

Вариант № 4

9.9880, 10.1865, 9.7946, 9.9539, 9.8977, 10.1190, 10.0806, 9.8619, 9.9394, 10.1472, 10.0553, 10.1535, 10.0505, 9.9927, 9.9318, 9.8900, 10.0663, 9.9153, 10.0316, 9.9738, 9.9660, 9.8653, 10.1018, 9.9610, 9.9454, 9.8860, 10.0469, 10.0011, 9.8799, 10.0129, 9.9789, 10.0056, 10.0234, 9.8766, 9.9053.

Вариант № 5

9.8643, 9.9669, 10.0635, 10.0931, 9.9096, 9.9156, 10.0708, 9.8872, 9.9399, 10.0133, 9.9507, 10.0289, 9.9355, 9.8944, 10.1479, 10.1038, 10.0485, 9.8996, 9.7995, 10.1237, 10.1957, 9.9565, 10.1553, 9.8681, 9.9626, 10.0535, 9.9739, 9.9935, 10.0371, 9.9921, 9.8814, 9.8783, 10.0056, 10.0021, 9.9796.

Вариант № 6

9.9571, 9.8420, 10.2112, 10.0656, 10.0036, 9.9372, 9.8708, 10.1138, 10.0498, 9.9921, 9.9754, 9.9535, 9.8787, 10.0232, 9.9959, 9.9850, 9.9146, 9.9010, 10.0290, 9.8975, 9.8651, 10.0728, 9.8884, 10.0953, 10.1488, 9.9405, 10.0551, 9.9632, 9.9679, 10.0086, 10.0462, 9.8832, 10.1340, 10.1593, 9.9316.

Вариант № 7

6.4043, 6.3997, 6.3883, 6.3989, 6.4118, 6.3978, 6.3846, 6.4030, 6.4047, 6.3982, 6.3926, 6.3993, 6.4008, 6.4083, 6.3929, 6.4099, 6.3942, 6.3897, 6.4077, 6.3916, 6.4060, 6.4050, 6.4004, 6.3934, 6.4017, 6.3948, 6.4055, 6.4021, 6.3974, 6.4015, 6.4040, 6.3956, 6.4069, 6.3964, 6.4000, 6.4023, 6.4037.

Вариант № 8

6.4083, 6.3929, 6.4048, 6.3996, 6.3917, 6.3848, 6.4044, 6.4006, 6.4056, 6.3899, 6.3974, 6.4031, 6.4088, 6.3930, 6.4063, 6.3934, 6.4009, 6.4153, 6.3952, 6.4101, 6.3942, 6.3970, 6.4002, 6.3979, 6.4075, 6.4040, 6.3990, 6.4022, 6.3959, 6.3984, 6.3999, 6.4015, 6.4027, 6.4040, 6.4051, 6.3888, 6.4018.

Вариант № 9

6.4041, 6.3929, 6.4040, 6.4090, 6.3888, 6.4036, 6.3975, 6.3869, 6.4083, 6.3944, 6.4076, 6.4020, 6.3920, 6.4057, 6.3981, 6.4050, 6.3931, 6.3935, 6.4066, 6.4114, 6.3972, 6.3997, 6.4027, 6.4047, 6.3991, 6.3906, 6.4012, 6.4023, 6.4008, 6.3955, 6.4016, 6.3999, 6.4051, 6.3961, 6.4017.

Вариант № 10

201.2870, 198.1963, 199.7649, 200.3568, 201.0156, 201.1718, 200.0889, 198.4153, 199.4338, 201.7864, 200.4416, 199.9436, 198.7515, 201.6425, 197.6094, 199.8565, 199.5232, 198.5751, 199.2220, 200.0001, 200.7931, 201.4795, 203.2748, 198.4884, 201.0715, 198.8869, 200.1777, 199.0468, 199.6165, 202.1227, 200.3181, 200.4911, 200.8567, 200.6423, 200.9217.

Вариант № 11

199.7954, 201.0350, 201.3435, 198.2183, 199.4384, 200.1320, 200.9349, 198.4840, 200.0322, 200.8534, 198.7674, 200.8668, 199.9703, 196.6965, 198.5793, 200.4722, 201.0887, 201.6002, 201.8810, 198.4941, 198.9623, 200.3860, 199.1175, 199.3512, 199.5562, 200.6655, 199.6490, 199.9112, 197.1890, 200.1880, 200.3290, 202.1653, 200.5705, 201.1986, 201.7837.

Вариант № 12

200.7616, 199.4002, 199.4735, 198.4863, 201.2234, 199.7199, 201.0679, 201.9354, 198.2803, 200.1709, 200.8534, 198.6171, 200.8869, 197.5016, 201.7863, 199.1606, 198.5119, 200.4315, 201.4113, 201.0964, 200.2620, 202.5386, 199.0346, 199.6016, 199.8028, 201.0029, 200.3511, 198.7924, 197.8386, 199.9277, 200.0605, 199.9707, 200.4909, 200.5848, 201.6234.

Вариант № 13

0.0481, 0.0461, 0.0492, 0.0502, 0.0511, 0.0430, 0.0460, 0.0488, 0.0453, 0.0497, 0.0521, 0.0519, 0.0531, 0.0466, 0.0463, 0.0500, 0.0436, 0.0528, 0.0490, 0.0464, 0.0479, 0.0486, 0.0509, 0.0544, 0.0515, 0.0475, 0.0504, 0.0516, 0.0438, 0.0537, 0.0470, 0.0482, 0.0484, 0.0498, 0.0494.

Вариант № 14

0.0480, 0.0498, 0.0514, 0.0517, 0.0563, 0.0467, 0.0473, 0.0459, 0.0509, 0.0522, 0.0482, 0.0463, 0.0481, 0.0501, 0.0465, 0.0460, 0.0464, 0.0498, 0.0556, 0.0496, 0.0507, 0.0469, 0.0475, 0.0540, 0.0485, 0.0490, 0.0515, 0.0528, 0.0534, 0.0440, 0.0487, 0.0493, 0.0489, 0.0520, 0.0531.

Вариант № 15

0.0467, 0.0492, 0.0502, 0.0430, 0.0520, 0.0481, 0.0459, 0.0511, 0.0514, 0.0466, 0.0490, 0.0515, 0.0529, 0.0460, 0.0546, 0.0474, 0.0548, 0.0487, 0.0463, 0.0498, 0.0464, 0.0469, 0.0509, 0.0481, 0.0479, 0.0482, 0.0486, 0.0559, 0.0494, 0.0499, 0.0496, 0.0504, 0.0532, 0.0517, 0.0524.

Задание № 2. К разделу 1.2

Создать исходные данные на основании функции, диапазона и шага (шаг – 0,1). Вывести на экран данные в виде таблицы. Изобразить на графике табличные данные. Выполнить аппроксимацию точек методом наименьших квадратов, линейной интерполяцией, интерполяцией Лагранжа и сплайном.

Вариант	Функция	Интервал
1	$y(t) = e^t$	[0.1; 6]
2	$y(t) = \ln(t)$	[0.1; 6]
3	$y(t) = 10/t$	[0.1; 6]
4	$y(t) = 0.4t^3$	[0.1; 6]

Окончание таблицы

Вариант	Функция	Интервал
5	$y(t) = \sin(t)$	[−3.14; 3.14]
6	$y(t) = t\sin(t)$	[0; 6]
7	$y(t) = t^2\sin(t)$	[−3.14; 3.14]
8	$y(t) = t$	[0.1; 3.14]
9	$y(t) = t\ln(t)$	[0.1; 6]
10	$y(t) = 1/(1 - t)$	[−0.9; 1]
11	$y(t) = t \cdot 3^t$	[0; 6]
12	$y(t) = t^2 \cdot 2^t$	[−2; 1.5]
13	$y(t) = t/(t^2 - 1)$	[−3.14; 3.14]
14	$y(t) = \cos(t)$	[−3.14; 3.14]
15	$y(t) = t\cos(t)$	[−6; 6]

Задание № 3. К разделу 1.3

Построить функцию и найти количество локальных минимумов функций и их значения.

Вариант	Функция
1	$f(x)=\sin(x)+0.5, x\in\left(-\frac{4\pi}{3};\frac{4\pi}{3}\right)$
2	$f(x)=\ln\left(x^2-6x+10\right), x\in(0;5)$
3	$f(x)=x^4-3x^3, x\in(-2;3)$
4	$f(x)=\text{arctg}x+0.5\ln\left(1+x^2\right)$
5	$f(x)=x^4-2x^3+x, x\in(-3;3)$
6	$f(x)=\ln(cos0.5x)+x^2, x\in\left(-\frac{7\pi}{6};\frac{7\pi}{6}\right)$
7	$f(x) = \sin(x)x,\ x \in (-2\pi;\ 2\ \pi)$
8	$f(x) = \sin(x)x^2,\ x \in (-\pi;\ \pi)$

Окончание таблицы

Вариант	Функция
9	$f(x)=\cos(x)-0.5x, x\in\left(-\frac{3\pi}{2};\frac{3\pi}{2}\right)$
10	$f(x)=\cos(x), x\in(-\pi;\pi)$
11	$f(x)=x^2 2^x, x\in(-5;5)$
12	$f(x)=ln\left(x^2+1-x\right)+x, x\in(-3;2)$
13	$f(x) = 0.5\sin(x), x \in (-3; 3)$
14	$f(x)=\sin(lnx), x\in(0;2)$
15	$f(x)=x^5-x^2, x\in(0;2)$

Задание № 4. К разделу 2

Исследовать данные, построить графики и схемы, позволяющие получить информацию, и визуализировать данные, обучить модель регрессии или классификации, оценить качество модели. Оформить документацию на *github.com* (код, данные, отчет, презентация).

https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset.

https://www.kaggle.com/c/titanic/data.

https://www.kaggle.com/wenruliu/adult-income-dataset.

https://www.kaggle.com/vishalyo990/prediction-of-quality-of-wine.

https://www.kaggle.com/budincsevity/szeged-weather.

https://www.kaggle.com/hugodarwood/epirecipes.

https://www.kaggle.com/mhdzahier/travel-insurance.

https://www.kaggle.com/hussainaliarif/amazon-best-seller-june-2021-products.

https://www.kaggle.com/residentmario/ramen-ratings.

https://www.kaggle.com/astefopoulos/earthquakes-in-greece-19012018.

ЗАКЛЮЧЕНИЕ

В пособии приведены лишь некоторые возможные численные методы, которые наиболее часто необходимы для обработки экспериментальных результатов: поиск грубых промахов, анализ контрольных карт Шухарта, определение минимумов и аппроксимация функций.

Определение погрешности предсказаний моделей машинного обучения остается открытым вопросам в метрологии по сей день. В пособие приведены основные методики оценки точности и качества модели, результата ее предсказания. Особое внимание уделяется современной методологии ведения проектов Agile, ведению и стандартизированному оформлению документооборота в современной IT-индустрии.

Наиболее интересны рассмотренные вопросы для студентов технического профиля. Для закрепления теоретических навыков предлагается решить задачи, предложенные в разделе 3.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ Р 8.736-2011 Государственная система обеспечения единства измерений. Измерения прямые многократные. Методы обработки результатов измерений. Основные положения.

2. *Горлач, Б. А.* Математическое моделирование. Построение моделей и численная реализация: учеб. пособие. / Б. А. Горлач, В. Г. Шахов. М.: Лань, 2021. – 292 с.

3. *Окрепилов, В. В.* Основы метрологии: учеб. / В. В. Окрепилов, Ю. А. Антохина, А. А. Оводенко и др. – СПб: ГУАП, 2020. – 476 с.

4. *Демидович, Б. П.* Основы вычислительной математики / Б. П. Демидович, И. А. Марон. – М.: Государственное издательство физико-математической литературы, 1963.

5. *Бахвалов, Н. С.* Численные методы / Н. С. Бахвалов, А. А. Корнев, Е. В. Чижонков. – М.: Лаборатория знания, 2021. – 636 с.

6. *Зорич, В. А.* Математический анализ / В. А. Зорич. – М.: МЦНМО, 2020.

7. *Гасников, А. В.* Современные численные методы оптимизации. Метод универсального градиентного спуска: уч. пособие. / А. В. Гасников. – М.: МФТИ, 2018. – 291 с.

8. *Рашка, С.* Python и машинное обучение / С. Рашка. – ДМК Пресс, 2017. – 418 с.

9. *Мюллер, А.* Введение в машинное обучение с помощью Python / А. Мюллер, С. Гвидо. – Вильямс, 2017. – 285 с.

СОДЕРЖАНИЕ

Учебное издание

Степашкина Анна Сергеевна

ЧИСЛЕННЫЕ МЕТОДЫ И МАШИННОЕ ОБУЧЕНИЕ В МЕТРОЛОГИИ

Учебное пособие

ISBN: 978-5-8088-1656-5

9 785808 816565

Публикуется в авторской редакции

Компьютерная верстка *И. А. Мосиной*

Подписано к печати 25.11.21. Формат 60 × 84 1/16.
Усл. печ. л. 2,6. Уч.-изд. л. 2,8.
Тираж 50 экз. Заказ № 530.

Редакционно-издательский центр ГУАП
190000, Санкт-Петербург, Б. Морская ул., 67