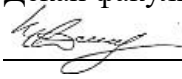
МИНОБРНАУКИ РОССИИ
Ярославский государственный университет им. П.Г. Демидова

Кафедра информационных и сетевых технологий

УТВЕРЖДАЮ
Декан факультета ИВТ
 Д.Ю. Чалый
«_23_»___мая_____2023 г.

Программа учебной практики
Технологическая (проектно-технологическая) практика

Направление подготовки
01.03.02 Прикладная математика и информатика

Направленность (профиль)
«Искусственный интеллект»

Квалификация выпускника
Бакалавр

Форма обучения
очная

Программа рассмотрена на заседании кафедры от 11 апреля 2023 г., протокол № 7

Программа одобрена НМК факультета ИВТ протокол № 6 от 28 апреля 2023 г.

Ярославль

1. Цели освоения дисциплины

Прохождение учебной практики: технологическая (проектно-технологическая) практика имеет цель формирование способности:

- управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни;
- разрабатывать и применять методы машинного обучения для решения практических задач;
- использовать инструментальные средства для решения задач машинного обучения;
- осуществлять сбор и подготовку данных для систем искусственного интеллекта.

При этом решаются задачи:

- закрепления и расширения знаний, полученных при изучении базовых дисциплин первого и второго курсов, повышения общей и профессиональной эрудиции;
- сбора и анализа теоретического и справочного материала для выполнения индивидуального задания;
- изучения языка R и среды разработки RStudio Cloud;
- осуществления поиска данных, их подготовки и разметки для выполнения практического задания;
- выбора методов машинного обучения для выполнения практического задания;
- определения метрики оценки результатов использования методов машинного обучения в рамках выполнения практического задания;
- выбора инструментальных средств для выполнения практического задания;
- разработки модели машинного обучения для выполнения практического задания;
- формулирования теоретических и практических выводов на основе критического переосмысления накопленного опыта.

2. Место дисциплины в структуре образовательной программы бакалавриата *(магистратуры, специалитета)*

Учебная практика: технологическая (проектно-технологическая) практика является обязательным элементом учебного процесса подготовки бакалавров по направлению 01.03.02 Прикладная математика и информатика, проводится в четвертом семестре.
Учебная практика: технологическая (проектно-технологическая) практика базируется на знаниях, полученных при изучении следующих модулей учебного плана: «Математика»; «Дискретная математика»; «Аппаратное и программное обеспечение компьютера», а также при изучении дисциплин: «Алгоритмы и алгоритмические языки»; «Структуры и алгоритмы обработки данных»; «Машинное обучение».
Результаты прохождения практики востребованы в ходе производственной практики и при подготовке выпускной квалификационной работы.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы бакалавриата *(магистратуры, специалитета)*

Процесс изучения дисциплины направлен на формирование следующих компетенций в соответствии с ФГОС ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции[1] (код и формулировка)**	**Перечень планируемых результатов обучения**
Универсальные компетенции		
УК-6. Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни.	ИУК6.1 Применяет знания о своих ресурсах и их пределах (личностных, психофизиологических, ситуативных, временных) для успешного выполнения порученной работы.	
	ИУК6.2 Планирует и реализует намеченные цели с учетом условий, средств, личностных возможностей, этапов карьерного роста, требований рынка труда.	
	ИУК6.3 Использует предоставленные возможности для приобретения новых знаний, умений и навыков.	
	ИУК6.4 Критически оценивает эффективность использования времени и других ресурсов при решении поставленных задач и относительно полученного результата.	
Профессиональные компетенции		

[1] Для образовательных программ, реализуемым в соответствии с ФГОС ВО, актуализированными с учетом профессиональных стандартов

ПК-4. Способен разрабатывать и применять методы машинного обучения для решения задач.	ИПК4.1 Проводит анализ требований и определяет необходимые классы задач машинного обучения.	Знать: Уметь: Владеть навыками:
	ИПК4.2 Определяет метрики оценки результатов моделирования и критерии качества построенных моделей.	Знать: Уметь: Владеть навыками:
	ИПК4.3 Принимает участие в оценке, выборе и при необходимости разработке методов машинного обучения.	
ПК-5. Способен использовать инструментальные средства для решения задач машинного обучения.	ИПК5.1 Осуществляет оценку и выбор инструментальных средств для решения поставленной задачи.	
	ИПК5.2 Разрабатывает модели машинного обучения для решения задач.	
	ИПК5.3 Создает, поддерживает и использует системы искусственного интеллекта, включающие разработанные модели и методы, с применением выбранных инструментов машинного обучения.	

ПК-7. Способен осуществлять сбор и подготовку данных для систем искусственного интеллекта.	ИПК7.1 Осуществляет поиск данных в открытых источниках, специализированных библиотеках и репозиториях.	
	ИПК7.2 Выполняет подготовку и разметку структурированных и неструктурированных данных для машинного обучения.	

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 6 зачетных единиц, 216 акад. часов.

Содержание разделов дисциплины:

Раздел 1. Подготовительный этап.
Подготовка и проведение установочного собрания. Ознакомление с приказом практики, с целью, задачами практики, консультации по выполнению содержания практики и заполнению отчетной документации. Составление индивидуального плана-графика прохождения практики.

Раздел 2. Ознакомительный этап.
Изучение возможностей языка R и среды разработки RStudio Cloud: организация работы в RStudio Cloud; операторы и работа с функциями в языке R; структуры данных в R; управление данными в R; визуализация данных в R.

Раздел 3. Практический этап.
Поиск, систематизация и обобщение необходимой для выполнения индивидуального задания научно-технической информации и литературы с использованием ресурсов и сервисов сети интернет и других источников. Выполнение индивидуальных заданий по поиску, подготовке и разметке данных, разработке модели машинного обучения для решения практической задачи.

Раздел 4. Заключительный.
Оформление документации по результатам практики. Подготовка и проведение итоговой аттестации по практике.

5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

Конкретные виды деятельности по каждому разделу практики и их продолжительность определяются индивидуально для каждого студента руководителем практики и/или научным руководителем.

Основными формами деятельности при прохождении им практики являются самостоятельная работа и консультации с руководителем практики и научным руководителем. Контроль выполнения разделов (этапов) практики осуществляет индивидуальный руководитель практики. Формой итоговой отчетности по практике является отчет о результатах выполнениязаданий по практике. Бланк отчета по практике приведен в приложении.

6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине

1. Microsoft Windows
2. RStudio Cloud

7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)

1. Браузер

8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», рекомендуемых для освоения дисциплины

а) основная литература

1. Бретт Ланц. Машинное обучение на R: экспертные техники для прогностического анализа. - Санкт-Петербург : Питер, 2020. - 464 с. - ISBN 978-5-4461-1512-9. - URL: https://ibooks.ru/bookshelf/367984/reading (дата обращения: 08.06.2022). - Текст: электронный.
2. Кун Макс. Предиктивное моделирование на практике. - Санкт-Петербург : Питер, 2019. - 640 с. - ISBN 978-5-4461-1039-1. - URL: https://ibooks.ru/bookshelf/365287/reading (дата обращения: 08.06.2022). - Текст: электронный.

б) дополнительная литература

1. Зададаев, С. А. Математика на языке R : учебник : [16+] / С. А. Зададаев. – Москва : Прометей, 2018. – 324 с. : схем., ил. – (Учебники для вузов. Специальная литература). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=494941 (дата обращения: 08.06.2022). – ISBN 978-5-907003-59-0. – Текст : электронный.
2. Воронина, В. В. Теория и практика машинного обучения : учебное пособие / В. В. Воронина. — Ульяновск : УлГТУ, 2017. — 290 с. — ISBN 978-5-9795-1712-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/165053
3. Казаченок, Н. Н. Учебная практика : учебно-методическое пособие / Н. Н. Казаченок, О. П. Михеева. — Тольятти : ТГУ, 2018. — 37 с. — ISBN 978-5-8259-1385-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/140139

в) ресурсы сети «Интернет»

1. Электронная библиотека «Университетская библиотека online». URL: http://biblioclub.ru/
2. Информационная система «Единое окно доступа к образовательным ресурсам». URL: http://window.edu.ru/
3. Образовательный портал Череповецкого государственного университета. URL: https://edu.chsu.ru/

Приложение № 1 к рабочей программе дисциплины
«Технологическая (проектно-технологическая) практика»

Фонд оценочных средств
для проведения текущего контроля успеваемости
и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания и иные материалы,
используемые в процессе текущего контроля успеваемости

Задание 1. Составьте индивидуальный плана-график прохождения практики. Форма представления результата: план-график прохождения практики.

Задание 2. Организуйте поиск, выполните систематизацию и обобщение научно-технической информации и литературы, необходимой для выполнения индивидуального задания. Форма представления результата: список литературы.

Задание 3. Изучите язык R и особенности среды разработки RStudio Cloud. Форма представления результата: приведено решение задач, направленных на изучение языка R и среды разработки RStudio Cloud.

Задание 4. Осуществите поиск данных, их подготовку и разметку для выполнения индивидуального задания. Форма представления результата: описание датасета.

Задание 5. Выберите методы машинного обучения для выполнения индивидуального задания.
Форма представления результата: описание методов машинного обучения, выбранных для решения индивидуального задания.

Задание 6. Определите метрики оценки результатов использования методов машинного обучения в рамках индивидуального задания.
Форма представления результата: описание метрик оценки результатов использования методов машинного обучения в рамках индивидуального задания.

Задание 7. Выберите инструментальные средства для выполнения индивидуального задания.
Приведите обоснование своего выбора.
Форма представления результата: описание инструментальные средства для выполнения индивидуального задания с обоснование выбора.

Задание 8. Разработайте модели машинного обучения для выполнения индивидуального задания.
Форма представления результата: описание модели машинного обучения для выполнения индивидуального задания.

Задание 9. Сформулируйте теоретические и практические выводы, а также результаты выполнения индивидуального задания на основе критического переосмысления накопленного опыта.
Форма контроля: теоретические и практические выводы

Примеры задач, направленных на изучение языка R и среды разработки RStudio Cloud:
Задание 1. Первый сеанс RStudio Cloud.

Выполните задания и ответьте на вопросы. Отчет о проделанной работе представьте в разделе отчета по практике «Язык R и среда разработки RStudio Cloud».

1. Запустите RStudio Cloud (https://rstudio.cloud). Пройдите регистрацию.
2. В своем рабочем пространстве создайте новый проект PracticeTask1.
3. Создайте простой набор данных (в терминологии R — вектор), состоящий из чисел 1, 2 и
   4, и присвойте ему имя x:

```
> x <- c(1,2,4)
```

Стандартным оператором присваивания в R является оператор <-. Также можно использовать оператор =, но лучше этого не делать, так как он не будет работать в некоторых специфических ситуациях.

Обратите внимание: с переменными не связываются никакие фиксированные типы.

Имя c означает конкатенацию (concatenate). В данном случае конкатенация выполняется с числами 1, 2 и 4. Точнее говоря, мы выполняем конкатенацию трех одноэлементных векторов, каждый из которых содержит одно из этих чисел. Дело в том, что любое число также рассматривается как вектор с одним элементом.

А теперь убедимся в том, что данные действительно хранятся в x. Чтобы вывести вектор на экран, просто введите его имя. При вводе любого имени переменной (или в более широком смысле — любого выражения) в интерактивном режиме R выведет значение этой переменной (или выражения).

Также для вывода можно использовать функцию print. Выведите на экран фразу «Это мой первый сеанс работы с R». Строковые выражения в R заключаются в кавычки.

4. Выполните команду, которая присваивает q значение (1,2,4,1,2,4,8), используя ранее определенный вектор x.

```
> q <- c(x,x,8)
```

5. Выведите на экран значение 3-го элемента вектора x.

Для обращения к отдельным элементам вектора используется синтаксис []. Как и в других языках, селектор (в данном случае 3) называется индексом. Индексирование элементов векторов R начинается с 1.

```
> x[3]
```

6. Выведите на экран элементы вектора x со 2 по 3.

Сегментация — одна из важнейших операций с векторами. Выражение x[2:3] обозначает подвектор x, состоящий из элементов с 2 по 3 (в данном случае элементы со значениями 2 и 4).

```
> x[2:3]
```

7. Выведите на экран математическое ожидание и среднеквадратическое отклонение для элементов вектора x.

```
> mean(x)
[1] 2.333333
```

```
> sd(x)
[1] 1.527525
```

В первой строке выражением является вызов функции mean(x). Возвращаемое значение этой функции выводится автоматически, вызывать функцию R print() для этого не обязательно. Среднеквадратическое отклонение рассчитывается при помощи функции sd(x).

8. Выведите на экран перечень внутренних наборов данных, предназначенных для демонстраций.

```
> data()
```

9. Вычислите математическое ожидание и среднеквадратическое отклонение для набора данных Nile (этот набор содержит данные о течении Нила).

```
> mean(Nile)
[1] 919.35
> sd(Nile)
[1] 169.2275
```

10. Выведите гистограмму этих данных:

```
> hist(Nile)
```

11. Сохраните проект.
12. Ответьте на вопросы:
    1) Как сформировать вектор?
    2) Что означает операция с (concatenate)?
    3) Как вывести на экран значение переменной величины?
    4) Нужно ли указывать тип переменной величины?
    5) Как вывести на экран перечень демонстрационных наборов данных?
    6) Как рассчитать среднеквадратическое отклонение?
    7) Как рассчитать математическое ожидание? 8) Как построить гистограмму?

Задание 2. Работа с функциями в языке R.

Выполните задания и ответьте на вопросы. Отчет о проделанной работе представьте в разделе отчета по практике «Язык R и среда разработки RStudio Cloud».

1. Запустите RStudio Cloud. Создайте новый проект PracticeTask2 в своем рабочем пространстве.
2. Определите функцию с именем oddcount(), предназначенную для подсчета нечетных чисел в целочисленном векторе:

```
# Функция подсчитывает количество нечетных целых чисел в
x  oddcount <- function(x) {   k <- 0 #в начале вычислений
количество нечетных чисел равно 0   for (n in x) {
   if (n %% 2 == 1) {# %% - вычисление
остатка      k <- k+1
   }
}
```

```
return
(k)
}
```

3. Выполните команду Code – Source File...
4. В режиме интерпретатора вычислите количество нечетных чисел в сформированных в предыдущей работе наборов данных x и q с использованием созданной функции:

```
> oddcount(x)
[1] 1
> oddcount(q)
[1] 2
```

5. Попытайтесь вывести в интерпретаторе значения переменных k и n. Что Вы получили?

```
> k
> n
```

Переменная, видимая только в теле функции, называется локальной по отношению к этой функции. В oddcount() k и n являются локальными переменными. Они пропадают после того, как функция вернет управление.

Формальные параметры в функции R являются локальными переменным. Чтобы вычислить результат вызова функции, R копирует каждый фактический аргумент в соответствующую переменную локального параметра, и изменения в этой переменной не будут видны за пределами функции.

Переменные, созданные за пределами функций, являются глобальными; они также доступны внутри функций.

6. Определите функцию sum_2, складывающую два числа, одно из которых имеет значение по умолчанию, равное 5. Протестируйте работу функции. Сохраните файл в своей папке под именем lab2_2. Значение по умолчанию задается при описании аргумента после знака равенства. Например, y=5.

```
> sum_2(3)
  [1]       8
> sum(3,8)
[1] 11
```

7. Напишите функцию fx, определенную на всей числовой прямой:

$$f(x) = \begin{cases} 1-(x+2)^2, & \text{при } x \le -2 \\ -\frac{x}{2}, & \text{при } -2 < x \le 2 \\ (x-2)^2+1, & \text{при } 2 < x \end{cases}$$

Протестируйте работу функции. Сохраните файл в своей папке под именем lab2_3.

```
> fx(4.5)
  [1]       7.25
> fx(-4.5)
[1] -5.25
```

```
> fx(1)
[1] -0.5
```

8. Сохраните проект.
9. Ответьте на вопросы:
   [1] Как определить функцию?
   [2] Как описать тело функции?
   [3] Как записывается оператор if в языке R?
   [4] Как записывается оператор for в языке R?
   [5] Какие переменные являются глобальными?
   [6] Какие переменные являются локальными по отношению к функции?
   [7] Как определить значения по умолчанию для аргументов функции?
   [8] В каком порядке должны располагаться аргументы функции, не имеющие значений по умолчанию, и аргументы, имеющие значения по умолчанию? **Задание 3.**

Структуры данных в R.

Изучите представленную теорию, выполните задания. Отчет о проделанной работе представьте в разделе отчета по практике «Язык R и среда разработки RStudio Cloud».

В разных языках программирования существует множество типов структур данных, каждая из которых имеет свои сильные и слабые стороны и подходит для определенного круга задач. Поскольку R — это язык программирования, широко применяемый для статистического поиска закономерностей, используемые им структуры данных разработаны именно для этого типа задач. Структуры данных R, наиболее часто встречающиеся в машинном обучении, — это векторы (vectors), факторы (factors), списки (lists), массивы (arrays), матрицы (matrices) и таблицы (фреймы) данных (data frames). Каждая из этих структур адаптирована к конкретной задаче управления данными, поэтому важно понять, как они будут взаимодействовать в R-проекте.

Векторы

Основной структурой данных в R является вектор. В нем хранится упорядоченное множество значений, называемых элементами. Вектор может содержать любое количество элементов. Однако все элементы вектора должны быть одного типа; например, вектор не может содержать и цифры, и текст. Тип вектора v определяется с помощью команды typeof(v). В машинном обучении обычно используются следующие типы векторов: integer (числа без десятичных знаков), double (числа со значениями после запятой), character (текстовые данные) и logical (принимающие значения TRUE или FALSE). Есть также два специальных значения: NA — отсутствующее (пропущенное или недоступное) значение и NULL — нет значения (пустое множество). Два последних значения могут показаться синонимичными, однако на самом деле они различаются. Значение NA является заполнителем для чего-либо другого, и поэтому его длина равна единице, а значение NULL — действительно пустое, и его длина равна нулю.
Вводить большое количество данных вручную утомительно, однако простые векторы можно создавать с помощью функции объединения c(). Вектору также можно присвоить имя с помощью оператора <. Это оператор присваивания языка R, используемый практически так же, как оператор = во многих других языках программирования. Для примера построим набор векторов, содержащих данные о трех пациентах. Мы создадим символьный вектор с названием subject_name для хранения имен трех пациентов, числовой вектор temperature для хранения температуры тела каждого из пациентов (в градусах Фаренгейта) и логический вектор с названием flu_status для хранения диагноза

каждого пациента (TRUE, если у него грипп, и FALSE — если нет). Эти три вектора выглядят так, как показано в следующем фрагменте кода:

```
subject_name <- c("John Doe", "Jane Doe", "Steve Graves")
temperature <- c(98.1, 98.6, 101.4)
flu_status <- c(FALSE, FALSE, TRUE)
```

Значения в R-векторах сохраняют свою последовательность. Поэтому для получения доступа к данным каждого пациента можно использовать его положение в наборе, начиная с 1, и указывать это число в квадратных скобках ([и]) после имени вектора. Например, чтобы получить значение температуры для пациентки Джейн Доу, второй в списке, достаточно ввести:

```
> temperature[2]
[1] 98.6
```

В языке R есть множество методов извлечения данных из векторов. Так, с помощью оператора двоеточия можно получить диапазон значений. Например, чтобы получить температуру тела второго и третьего пациентов, введите следующее:

```
> temperature[2:3]
[1]  98.6 101.4
```

Чтобы исключить вывод элемента вектора, нужно указать его номер со знаком «минус». Так, для исключения данных о температуре второго пациента введите:

```
> temperature[-2]
[1]  98.1 101.4
```

Иногда полезно указать логический вектор, каждый элемент которого говорит о том, следует ли включать во множество результатов соответствующий элемент. Например, чтобы включить два первых показания температуры, но исключить третье, введите следующее:

```
> subject_name[c(TRUE, TRUE, FALSE)]
[1] "John Doe" "Jane Doe"
```

Вектор является основой для многих других структур данных R. Поэтому знание различных векторных операций имеет решающее значение для работы с данными в R.

Факторы

Номинальные признаки представляют собой характеристику с категориями значений. Для хранения номинальных данных можно использовать вектор символов, однако в R есть структура данных, специально предназначенная для этой цели. Фактор — это особый случай вектора, который нужен исключительно для представления категориальных или порядковых переменных. В медицинском наборе данных, который мы создаем, можно использовать фактор для указания пола пациента, для этого есть две категории: мужчины и женщины. Зачем использовать факторы, а не векторы символов? Одним из преимуществ факторов является то, что метки категорий сохраняются только один раз. Вместо сохранения слов MALE, MALE и FEMALE компьютер сохранит числа 1, 1 и 2, что позволит сократить объем памяти, необходимый для хранения значений. Кроме того, разные алгоритмы ML по-разному обрабатывают номинальные и числовые функции. Кодирование категориальных переменных как факторов гарантирует, что R будет обрабатывать категориальные данные надлежащим образом.
Для того чтобы создать фактор из символьного вектора, просто примените функцию factor(), например:

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
```

Обратите внимание, что при отображении фактора gender выводится дополнительная информация о его уровнях. Уровни (levels) представляют собой множество возможных категорий, которые может принимать фактор, в данном случае MALE и FEMALE. Создавая факторы, можно добавлять в них дополнительные уровни, которые могут быть

не представлены в исходных данных. Предположим, мы создали еще один фактор для группы крови, как показано в следующем примере:

```
> blood <- factor(c("O", "AB", "A"), levels = c("A", "B", "AB", "O"))
> blood
[1] O  AB A
Levels: A B AB O
```

Определив фактор blood, с помощью параметра levels мы указали дополнительный вектор, состоящий из четырех возможных групп крови. Несмотря на то, что данные включают в себя только группы крови O, AB и A, в факторе blood, как показывают результаты, сохраняются все четыре типа. Сохранение дополнительного уровня позволит добавлять пациентов с другой группой крови. Это также гарантирует, что, если бы мы создали таблицу групп крови, мы бы знали, что тип B существует, несмотря на его отсутствие в наших исходных данных. Фактор как структура данных дает возможность включать информацию о последовательности категорий номинальных переменных, что позволяет создавать порядковые признаки. Например, предположим, что у нас есть данные о выраженности симптомов у пациента, закодированные в порядке возрастания степени тяжести — от легкой до умеренной или тяжелой. Мы указываем наличие порядковых данных, предоставляя уровни фактора в желаемой последовательности, перечисляя их в порядке возрастания — от самого низкого до самого высокого — и присваивая параметру ordered значение TRUE, как показано ниже:

```
> symptoms <- factor(c("SEVERE", "MILD", "MODERATE"), levels = c("MILD", "MODERATE", "SEVERE"), ordered = TRUE)
> symptoms
[1] SEVERE   MILD     MODERATE
Levels: MILD < MODERATE < SEVERE
```

Полученный в результате фактор symptoms теперь включает в себя информацию о требуемом порядке. В отличие от предыдущих факторов уровни этого фактора разделены символами < для обозначения последовательности от MILD до SEVERE:
Полезным свойством упорядоченных факторов является то, что для них логические тесты работают так, как требуется. Например, можно проверить, насколько выражены симптомы каждого пациента — серьезнее ли они, чем умеренные:

```
> symptoms > "MODERATE"
[1]  TRUE FALSE FALSE
```

Алгоритмы машинного обучения, способные моделировать упорядоченные данные, ожидают на входе упорядоченные факторы, поэтому обязательно представляйте данные соответствующим образом.

Списки

Список — это структура данных, очень похожая на вектор тем, что она тоже используется для хранения упорядоченного множества элементов. Однако если для вектора требуется, чтобы все его элементы были одинакового типа, то список позволяет собирать разные типы данных. Благодаря такой гибкости списки часто используются для хранения различных типов входных и выходных данных, а также параметров конфигурации для моделей машинного обучения. Для того чтобы продемонстрировать списки, рассмотрим построенный нами набор данных о трех пациентах, который хранится в шести векторах. Если мы хотим отобразить все данные для первого пациента, то нам нужно ввести пять Rкоманд:

```
> subject_name [1]
[1] "John Doe"

> temperature [1]
[1] 98.1

> flu_status [1]
[1] FALSE

> gender [1]
[1] MALE
Levels: FEMALE MALE

> blood [1]
[1] O
Levels: A B AB O

> symptoms [1]
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

Чтобы иметь возможность исследовать данные о пациенте повторно, не вводя заново эти команды, можно использовать список. Он позволяет сгруппировать все значения в один объект, который можно использовать повторно. Подобно тому как вектор создается с помощью функции c(), список создается с помощью функции list(), как показано в следующем примере. Единственное заметное различие заключается в том, что при создании списка каждому компоненту в последовательности должно быть присвоено имя. Наличие имен впоследствии позволит получить доступ к значениям по имени, а не по порядковому номеру. Для того чтобы создать список с именованными компонентами для всех данных первого пациента, введите следующую команду:

```
> subject1 <- list(fullname = subject_name[1],
+                  temperature = temperature[1],
+                  flu_status = flu_status[1],
+                  gender = gender[1],
+                  blood = blood[1],
+                  symptoms = symptoms[1])
```

Теперь данные пациента собраны в списке subject1:

```
> subject1
$fullname
[1] "John Doe"

$temperature
[1] 98.1

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: FEMALE MALE

$blood
[1] O
Levels: A B AB O

$symptoms
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

Обратите внимание, что значения помечены именами, которые мы указали в предыдущей команде. Поскольку список сохраняет последовательность значений так же, как и вектор, к его компонентам можно получить доступ по порядковым номерам, как показано далее для значения temperature:

```
> subject1[2]
$temperature
[1] 98.1
```

Результатом применения векторных операторов к объекту списка является другой объект списка, который представляет собой подмножество исходного списка. Например, предыдущий код возвращает список, состоящий из единственного компонента temperature. Чтобы вернуть один элемент списка с его собственным типом данных, при выборе компонента списка используйте двойные скобки ([[и]]). Например, следующая команда возвращает числовой вектор единичной длины:

```
> subject1[[2]]
[1] 98.1
```

Для точности лучше обращаться к компонентам списка по имени, ставя после имени списка знак $ и имя компонента следующим образом:

```
> subject1$gender
[1] MALE
Levels: FEMALE MALE
```

Подобно записи в двойных скобках, эта запись возвращает компонент списка его собственного типа данных (в данном случае — числовой вектор единичной длины).
Для того чтобы получить несколько элементов списка, нужно указать вектор имен. Следующая команда возвращает подмножество списка subject1, который содержит только компоненты temperature и flu_status:

```
> subject1[c("temperature", "flu_status")]
$temperature
[1] 98.1

$flu_status
[1] FALSE
```

С помощью списков и списков списков можно строить целые наборы данных. Например, можно создать списки subject2 и subject3 и сгруппировать их в списочный объект с именем pt_data. Такое построение набора данных широко распространено, но в языке R есть специальная структура данных.

Фреймы данных

Наиболее важной структурой данных R, используемой в ML, является, безусловно, фрейм данных — структура, аналогичная электронной таблице или базе данных, в которой есть строки и столбцы. В контексте языка R фрейм данных можно представить как список векторов или факторов, каждый из которых имеет одинаковое количество значений. Поскольку фрейм данных является, в сущности, списком объектов векторного типа, он объединяет в себе свойства, как векторов, так и списков. Создадим фрейм данных для нашего набора данных о пациентах. Для этого используем созданные ранее векторы данных о пациенте и вызовем функцию data.frame(), которая объединит их во фрейм данных:

```
> pt_data <- data.frame(subject_name,
+                       temperature,
+                       flu_status,
+                       gender,
+                       blood,
+                       symptoms,
+                       stringsAsFactors = FALSE)
```

Как вы могли заметить, в этом коде появилось нечто новое. Мы включили сюда дополнительный параметр: stringsAsFactors = FALSE.
Если его не указать, то R автоматически преобразует каждый символьный вектор в фактор. В данном случае, например, поле subject_name определенно не считается категориальными данными, поскольку имена не являются категориями значений. Поэтому, присвоив параметру stringsAsFactors значение FALSE, мы будем преобразовывать символьные векторы в факторы только в тех случаях, когда это имеет смысл для проекта.
При выводе фрейма данных pt_data мы видим, что его структура сильно отличается от структур данных, с которыми мы уже имели дело:

```
> pt_data
  subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
2     Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
```

В отличие от одномерных векторов, факторов и списков фрейм данных имеет два измерения и отображается в матричном формате. Конкретно в этом фрейме данных есть один столбец для каждого вектора данных пациента и одна строка для каждого пациента. В машинном обучении столбцы фрейма данных — это признаки, или атрибуты, а строки — примеры. Для того чтобы извлечь целые столбцы (векторы) данных, можно воспользоваться тем, что фрейм данных представляет собой просто список векторов. Подобно спискам, самый прямой способ извлечь отдельный элемент — обратиться к нему по имени. Например, чтобы получить вектор subject_name, введите следующее:

```
> pt_data$subject_name
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

Подобно спискам, вектор имен можно использовать для извлечения нескольких столбцов из фрейма данных:

```
> pt_data[c("temperature", "flu_status")]
  temperature flu_status
1        98.1      FALSE
2        98.6      FALSE
3       101.4       TRUE
```

Если запрашивать столбцы во фрейме данных по имени, то результатом будет фрейм, содержащий все строки данных для указанных столбцов.
Команда pt_data[,2:3] тоже извлечет столбцы temperature и flu_status, однако обращение к столбцам по имени приводит к ясному и простому в сопровождении R-коду, который не перестанет работать, если фрейм данных будет впоследствии переупорядочен.

```
> pt_data[,2:3]
  temperature flu_status
1        98.1      FALSE
2        98.6      FALSE
3       101.4       TRUE
```

Для извлечения из фрейма данных конкретных значений используются методы, подобные тем, что применяются для доступа к значениям векторов. Однако здесь есть важное отличие: поскольку фрейм данных является двумерным, необходимо указывать не только нужные строки, но и столбцы. Сначала указываются строки, затем ставится запятая, за которой следуют столбцы: [row, col]. Как и для векторов, номера строк и столбцов начинаются с единицы. Например, чтобы извлечь значение, стоящее в первой строке и втором столбце фрейма данных пациента, используется следующая команда:

```
> pt_data[1, 2]
[1] 98.1
```

Если требуется извлечь несколько строк или столбцов данных, то нужно указать векторы, соответствующие желаемым строкам и столбцам. Следующий оператор извлекает данные из первой и третьей строк, а также из второго и четвертого столбцов:

```
> pt_data[c(1, 3), c(2, 4)]
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

Чтобы сослаться на каждую строку или столбец, просто оставьте часть строки или столбца пустой. Например, чтобы извлечь все строки первого столбца, введите команду:

```
> pt_data[, 1]
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

Так можно извлечь все столбцы для первой строки:

```
> pt_data[1, ]
  subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
```

А так — все данные:

```
> pt_data[ , ]
  subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
2     Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
```

Конечно, к столбцам лучше обращаться по имени, а не по порядковому номеру, а для исключения из вывода строк или столбцов данных можно использовать отрицательные значения. Таким образом, результат команды:

```
> pt_data[c(1, 3), c("temperature", "gender")]
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

эквивалентен результату следующей:

```
> pt_data[-2, c(-1, -3, -5, -6)]
  temperature gender
1        98.1   MALE
3       101.4   MALE
```

Иногда во фреймах данных необходимо создавать новые столбцы — например, как функцию от существующих столбцов. Так, нам может потребоваться преобразовать показания температуры во фрейме данных о пациенте из шкалы Фаренгейта в шкалу Цельсия. Для этого достаточно воспользоваться оператором присваивания и присвоить результат преобразования столбцу с новым именем следующим образом:

```
> pt_data$temp_c <-(pt_data$temperature - 32) * (5 / 9)
```

Чтобы убедиться в правильности вычислений, сравним новый столбец temp_c с температурой по Цельсию с предыдущим столбцом temperature, где указана температура по Фаренгейту:

```
> pt_data[c("temperature", "temp_c")]
  temperature   temp_c
1        98.1 36.72222
2        98.6 37.00000
3       101.4 38.55556
```

Увидев эти показания рядом, мы можем подтвердить, что расчет выполнен верно. Чтобы лучше ознакомиться с фреймами данных, попробуйте выполнить аналогичные операции с набором данных о пациентах или, что еще лучше, использовать данные одного из ваших собственных проектов.

Матрицы

Матрица — это структура данных, сведенных в двумерную таблицу, где данные представлены в виде строк и столбцов. Подобно векторам, матрицы в R могут содержать только один тип данных, хотя они чаще всего используются для математических операций, и поэтому обычно хранят только числа. Для того чтобы создать матрицу, достаточно вызвать функцию matrix() и указать вектор данных, а также параметр, определяющий количество строк (nrow) или столбцов (ncol). Например, для создания матрицы 2 × 2, в которой хранятся числа от единицы до четырех, можно использовать параметр nrow, чтобы запросить данные, которые будут разделены на две строки:

```
> m <- matrix(c(1, 2, 3, 4), nrow = 2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Или же можно создать такую же матрицу с использованием параметра ncol = 2:

```
> m <- matrix(c(1, 2, 3, 4), ncol = 2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Вероятно, вы заметили, что в R сначала загружается первый столбец матрицы, а затем — второй. Это называется размещением по столбцам, которое применяется в R по

умолчанию для загрузки матриц. Для размещения по строкам используется параметр byrow=T.

```
> m <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = T)
> m
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Рассмотрим на примере, что произойдет, если увеличить количество значений матрицы. Если указать шесть значений, то для двух строк будет создана матрица из трех столбцов:

```
> m <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Если задать два столбца, то будет создана матрица из трех строк:

```
> m <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
> m
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Как и в случае с фреймами данных, значения матриц могут быть извлечены с использованием нотации [row, column]. Например, m[1,1] для матрицы m вернет значение 1, а m[3,2] — 6. Кроме того, можно извлекать целые строки или столбцы:

```
> m[1, ]
[1] 1 4
> m[, 1]
[1] 1 2 3
```

Задания:

1. Запустите RStudio Cloud. В своем рабочем пространстве создайте новый проект PracticeTask3.
2. Создайте вектор a1 из вектора a1 получите вектор a1, добавив в середину вектора число 161.

```
> a1<-c(34,5,73,12)
> a1
[1] 34  5 73 12
```

3. Создайте фактор month1, содержащий названия весенних месяцев года, при этом должен быть задан полный перечень месяцев и установлен порядок следования месяцев.
4. Сформируйте список emp, содержащий сведения о сотруднике: имя (name – имеет символьные значения); зарплата (salary – вещественное число), принадлежность к профсоюзу (union – логическое значение). Внесите данные об одном сотруднике: Irina, 40800.00, F. Выведите зарплату сотрудника. Измените зарплату сотрудника, задав значение 50000.00.
5. Создайте фрейм department, описав сотрудников отдела атрибутами: name; salary; union. Введите данные по 2-м сотрудникам. Выведите имена всех сотрудников. Выведите зарплату второго сотрудника.
6. Создайте матрицу m размером 3x2.

```
> m
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

На основании матрицы m создайте новую матрицу m2 следующего вида:

```
> m2
     [,1] [,2]
[1,]    3    6
[2,]    2    5
```

7. Сохраните проект. **Задание 4. Управление данными в R.**

Изучите представленную теорию, выполните задания. Отчет о проделанной работе представьте в разделе отчета по практике «Язык R и среда разработки RStudio Cloud».

Одна из проблем, с которыми приходится сталкиваться при работе с большими наборами данных, заключается в сборе, подготовке данных, поступающих из различных источников, и управлении ими. Далее на примере реальных задач машинного обучения мы рассмотрим подготовку данных, их очистку и управление ими.

Сохранение, загрузка и удаление структур данных в R

Если вы один раз потратили много времени на преобразование фрейма данных в нужную форму, то вам не нужно будет заново все переделывать при каждом перезапуске R-сеанса. Чтобы сохранить структуру данных в файле, который впоследствии можно загрузить или перенести в другую систему, используйте функцию save(). Функция записывает одну или несколько структур данных R в местоположение, указанное параметром file. **Файлы данных R имеют расширение .RData**. Предположим, у вас есть три объекта с именами x, y и z, которые вы хотели бы сохранить в постоянном файле. Независимо от того, являются ли эти объекты векторами, факторами, списками или фреймами данных, их можно сохранить в файле mydata.RData с помощью следующей команды:

```
> save(x, y, z, file = "mydata.RData")
```

Задание: откройте файл PracticeTask3, созданный как результат выполнение предыдущего задания, сохраните объекты с именами subject1, department, month1 в файл mydata.RData.

Команда load() позволяет воссоздать любые структуры данных, которые были сохранены в файле .RData. Для того чтобы загрузить созданный файл mydata.RData, просто введите следующее:

```
> load("mydata.RData")
```

По этой команде в среде R будут воссозданы структуры данных x, y и z.

Задание: создайте новый файл с именем lab2.R и загрузите в него объекты из файла mydata_lab1.RData, выведите объекты subject1, department, month1.

Если вам нужно быстро завершить сеанс R, воспользуйтесь командой save.image() — она запишет весь сеанс в файл, называемый .RData. По умолчанию при следующем запуске R будет искать этот файл, и сеанс будет воссоздан точно в том виде, в каком вы его оставили. ***Задание***: сохраните текущий сеанс, воспользовавшись командой save.image().

После того как вы некоторое время поработаете в сеансе R, вы, возможно, накопите несколько структур данных. Функция ls() возвращает вектор, который состоит из всех

структур данных, находящихся в памяти в настоящий момент. Например, если вы выполняли все предыдущие команды задания, то функция ls() возвратит следующее:

```
> ls()
[1] "department" "month1"     "subject1"
```

При выходе из сеанса R автоматически удаляет из памяти все структуры данных, однако для больших объектов, возможно, вы захотите освободить память раньше. Для этого можно использовать функцию удаления rm(). Например, чтобы удалить объекты month1 и subject1, просто введите следующее:

```
> rm(month1, subject1)
> ls()
[1] "department"
```

Функции rm() также можно передать символьный вектор имен объектов, подлежащих удалению. Если же передать ей результат работы функции ls(), то будет очищен весь сеанс R:

```
> rm(list = ls())
> ls()
character(0)
```

Будьте очень ос торожны при выполнении этой команды, так как перед удалением объектов вы не получите предупреждения!

Задания:

1. Используя функцию rm() удалите объекты month1 и subject1, проверьте список оставшихся объектов, используя функцию ls().
2. Удалите все объекты.
3. Используя файл .RData, восстановите объекты.

Импорт и сохранение данных из CSV-файлов

Общедоступные наборы данных обычно хранятся в текстовых файлах. Текстовые файлы могут быть прочитаны практически на любом компьютере и в любой операционной системе, что делает этот формат универсальным. Текстовые файлы также могут быть экспортированы и импортированы в такие программы, как Microsoft Excel, что позволяет быстро и легко работать с данными, представленными в виде электронных таблиц. Табличный (представленный в виде таблицы) файл данных имеет матричную структуру: каждая строка текста отражает один пример и каждый пример имеет одинаковое количество признаков. Значения признаков в каждой строке разделены предопределенным символом, который называется разделителем. В первой строке файла табличных данных часто содержатся имена столбцов. Эта строка называется заголовком. Пожалуй, самым распространенным форматом табличного текстового файла является файл значений, разделенных запятыми (commaseparated values, CSV), в котором, как следует из названия, в качестве разделителя используются запятые. CSV-файлы можно импортировать и экспортировать во многих распространенных приложениях. CSV-файл, описывающий набор медицинских данных, может быть сохранен следующим образом:

```
subject_name,temperature,flu_status,gender,blood_type
John Doe,98.1,FALSE,MALE,O
Jane Doe,98.6,FALSE,FEMALE,AB
Steve Graves,101.4,TRUE, MALE,A
```

Пусть файл с данными о пациентах pt_data.csv расположен в рабочем каталоге R. Тогда функцию read.csv() можно использовать следующим образом:

```
> pt_data<-read.csv("pt_data.csv",stringsAsFactors = FALSE)
```

Эта функция прочитает CSV-файл и поместит данные во фрейм с именем pt_data. Подобно тому, как мы сделали это при создании фрейма данных, здесь нужно использовать параметр stringsAsFactors = FALSE, чтобы предотвратить преобразование всех текстовых переменных в факторы. Если вы уверены, что все столбцы в CSV-файле действительно

являются факторами, данный шаг лучше выполнить самостоятельно, а не предоставлять это R.
Если набор данных находится за пределами рабочего каталога R, то при вызове функции read.csv() нужно указать полный путь к CSV-файлу (например, "/path/to/mydata.csv").
По умолчанию R предполагает, что в CSV-файле есть строка заголовка, в которой перечислены имена объектов из набора данных. Если в CSV-файле нет заголовка, укажите параметр header = FALSE, как показано в следующей команде, и R назначит имена признаков по умолчанию, нумеруя их V1, V2 и т. д.:

```
> pt_data<-read.csv("pt_data.csv",stringsAsFactors = FALSE,header=FALSE)
```

Функция read.csv() является частным случаем функции read.table(), которая позволяет считывать табличные данные в различных форматах, включая другие форматы с разделителями, такие как значения, разделенные табуляцией (tab-separated values, TSV).
Чтобы сохранить фрейм данных в CSV-файле, используйте функцию write.csv(). Если фрейм данных называется pt_data, просто введите следующее:

```
> write.csv(pt_data, file="pt_data.csv",row.names = FALSE)
```

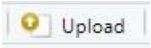
По этой команде в рабочий каталог R будет записан CSV-файл с именем pt_data.csv. Параметр row.names переопределяет заданное по умолчанию поведение R, которое заключается в выводе имен строк в CSV-файле. Как правило, эти выходные данные не нужны и будут лишь увеличивать размер получаемого файла.

Задания:

1. Откройте файл R, созданный в ходе выполнения предыдущего задания. Запишите в файл pt_data.csv фрейм данных pt_data, определив параметр row.names=FALSE.
2. Вернитесь в файл R текущей работы. Прочитайте содержимое файла pt_data.csv.

Исследование данных и их интерпретация

После сбора данных и загрузки их в структуры данных R следующий шаг в процессе машинного обучения — подробное исследование данных. Именно на этом этапе начинается исследование признаков и примеров данных, а также выявление особенностей, которые делают эти данные уникальными. Чем лучше будут интерпретированы данные, тем лучше вы сможете выбрать модель ML для вашей задачи.
Рассмотрим набор данных usedcars.csv, который содержит реальные данные о подержанных автомобилях, выставлявшихся на продажу на популярном американском сайте в 2012 году.

Задание: загрузите файл usedcars.csv на сервер, если Вы используете среду RStudio Cloud, используя кнопку 

.

Поскольку набор данных хранится в форме CSV, мы можем использовать функцию read.csv() для загрузки данных из него во фрейм данных R:

```
> usedcars <- read.csv("usedcars.csv", stringsAsFactors = FALSE)
```

Задание: загрузите данные их файла usedcars.csv во фрейм usedcars.

Теперь, когда у нас есть фрейм данных usedcars, представьте себя в роли специалиста по анализу данных, который должен изучить данные о подержанных автомобилях. Несмотря на то, что изучение данных — гибкий процесс, его этапы можно представить как своего рода исследование, в котором даются ответы на вопросы о данных. Вопросы могут

различаться в зависимости от проекта, однако типы вопросов всегда похожи. Вы сможете адаптировать основные этапы этого исследования к любому набору данных, будь он большим или маленьким. ***Структуры данных***

Первые вопросы, которые следует задать при исследовании нового набора данных, должны быть о том, как организован набор данных. Если повезет, то ваш источник предоставит словарь данных — документ, в котором описаны признаки набора данных. В нашем случае данные о подержанных автомобилях не сопровождаются таким документом, поэтому придется создать его самостоятельно. Функция str() представляет собой метод отображения структуры R-объектов, таких как фреймы данных, векторы и списки. Эту функцию можно использовать для создания базовой схемы словаря данных:

```
> str(usedcars)
'data.frame':   150 obs. of  6 variables:
 $ year        : int  2011 2011 2011 2011 2012 2010 2011 2010 2011 2010 ...
 $ model       : chr  "SEL" "SEL" "SEL" "SEL" ...
 $ price       : int  21992 20995 19995 17809 17500 17495 17000 16995 16995 16995 ...
 $ mileage     : int  7413 10926 7351 11613 8367 25125 27393 21026 32655 36116 ...
 $ color       : chr  "Yellow" "Gray" "Silver" "Gray" ...
 $ transmission: chr  "AUTO" "AUTO" "AUTO" "AUTO" ...
```

Задание: отобразите структуру usedcars, используя команду str().

Как много информации о наборе данных мы получили с помощью такой простой команды! Фрагмент 150 obs означает, что данные включают в себя 150 наблюдений — еще один способ сообщить, что набор данных содержит 150 записей, или примеров. Количество наблюдений часто обозначается буквой n. Поскольку мы знаем, что данные описывают подержанные автомобили, то можем теперь предположить, что у нас есть примеры n = 150 автомобилей для продажи. Фрагмент 6 variables говорит о наличии шести признаков, которые записаны в данных. Эти признаки перечислены по названиям в отдельных строках.

Глядя на строку для признака color, можно отметить дополнительные детали:

```
 $ color       : chr  "Yellow" "Gray" "Silver" "Gray" ...
```

После имени переменной стоит метка chr, которая сообщает, что этот признак является символьным. В этом наборе данных три переменные символьные, а три отмечены как int, что означает тип integer. Набор данных usedcars содержит только символьные и целочисленные переменные, однако в других случаях, при использовании не только целочисленных данных, можно встретить тип num или numeric. Все факторы будут иметь тип factor. После типа каждой R-переменной указана последовательность первых нескольких значений признаков. Первыми четырьмя значениями признака color являются "Yellow", "Gray", "Silver" и "Gray". Применив к именам и значениям признаков некоторые знания из предметной области, можно сделать предположения о том, что представляют собой переменные. Переменная year может указывать на год выпуска автомобиля или на год, когда было опубликовано объявление о продаже. Позже нужно будет исследовать этот признак более подробно, поскольку четыре примера значений (2011, 2011, 2011, 2011) могут послужить аргументом в пользу любой из этих версий. Переменные model, price, mileage, color и transmission относятся к характеристикам продаваемого автомобиля. Похоже, что данным были присвоены осмысленные имена переменных, однако это не всегда так. Иногда наборы данных имеют признаки с бессмысленными именами или кодами, такими как V1. В этих случаях может потребоваться дополнительное расследование, чтобы определить, что представляет собой тот или иной признак. Но даже с информативными именами признаков следует всегда скептически относиться к предоставленным меткам. Продолжим исследование.

Числовые переменные

Чтобы исследовать числовые переменные в данных о подержанных автомобилях, мы воспользуемся базовым набором измерений, описывающим значения, известные как сводная статистика. Функция summary() позволяет вывести несколько видов сводной статистики. Рассмотрим один из признаков — year:

```
> summary(usedcars$year)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2000    2008    2009    2009    2010    2012
```

Пока не будем обращать внимание на конкретные значения — важен тот факт, что здесь есть цифры 2000, 2008 и 2009, которые говорят о том, что переменная year указывает на год выпуска автомобиля, а не на год, когда было опубликовано объявление, поскольку, как известно, списки автомобилей были составлены в 2012 году. Если передать функции summary() вектор имен столбцов, то получим сводную статистику сразу для нескольких числовых переменных:

```
> summary(usedcars[c("price","mileage")])
     price          mileage
 Min.   : 3800   Min.   :  4867
 1st Qu.:10995   1st Qu.: 27200
 Median :13592   Median : 36385
 Mean   :12962   Mean   : 44261
 3rd Qu.:14904   3rd Qu.: 55124
 Max.   :21992   Max.   :151479
```

Сводная статистика, предоставленная функцией summary(), — простой, но мощный инструмент для исследования данных. Она бывает двух типов: меры центра и меры разброса.

> ***Задание***: используя функцию summary() запросите статистику по всем числовым переменным usedcars.

Измерение средних значений: среднее арифметическое и медиана

Измерение средних значений — это задача из статистики, используемая для определения значения, которое попадает в середину набора данных. Скорее всего, один распространенный показатель центра вам уже знаком: среднее значение. В целом, когда чтото считается средним, это значит, что оно находится где-то между крайними точками шкалы. Если у ученика средний балл успеваемости, это говорит о том, что ему были выставлены средние отметки в классе. В целом средний предмет типичен и не особо отличается от других в своей группе. Его можно представить как образец, по которому судят обо всех остальных. В статистике среднее также называется средним арифметическим — это измерение, определяемое как сумма всех значений, разделенная на количество значений. Например, для того, чтобы вычислить средний доход в группе из трех человек с доходами 36 000, 44 000 и 56 000 долларов, можно выполнить следующую команду:

```
> (36000+44000+56000)/3
[1] 45333.33
```

В R также ес ть функция mean(), которая вычисляет среднее арифметическое вектора чисел:

```
> mean(c(36000,44000,56000))
[1] 45333.33
```

> ***Задание***: используя функцию mean() посчитайте средние значения для всех числовых переменных usedcars.

Средний доход этой группы людей составляет примерно 45 333 доллара. Концептуально это можно представить как доход, который был бы у каждого человека, если бы общая

сумма дохода делилась поровну на всех. При выводе предыдущих результатов summary() были заданы средние значения переменных price и mileage. Эти значения указывают на то, что типичная подержанная машина в этом наборе данных стоит 12 962 доллара, а показание ее одометра — 44 261. Что это говорит о данных? Поскольку средняя цена относительно низкая, то можно ожидать, что в наборе данных содержатся автомобили экономкласса. Конечно, среди них могут также присутствовать автомобили повышенной комфортности последней модели с большим пробегом. Но сравнительно низкий статистический показатель среднего пробега не приводит доказательств в поддержку этой гипотезы. Впрочем, он также не предоставляет доказательств, позволяющих проигнорировать такую возможность. Следует помнить об этом при дальнейшем изучении данных. Среднее арифметическое является наиболее часто упоминаемым статистическим показателем для измерения среднего значения в наборе данных, однако не всегда самым подходящим. Другим часто используемым показателем среднего значения выступает медиана — значение, которое находится в середине упорядоченного списка значений. Как и в случае со средним арифметическим, в R есть функция median(), которую можно применить к нашим данным о зарплате следующим образом:

```
> median(c(36000,44000,56000))
[1] 44000
```

Итак, медианный доход составляет 44 000 долларов.

На первый взгляд кажется, что медиана и среднее арифметическое очень похожи. Конечно, среднее арифметическое 45 333 доллара и медиана 44 000 долларов не очень различаются. Зачем же нужны два измерения средних значений? Причина заключается в том, что среднее арифметическое и медиана по-разному зависят от значений, находящихся на границах диапазона. В частности, среднее арифметическое очень чувствительно к выбросам — значениям, которые нетипично велики или малы по сравнению с большинством данных. Таким образом, поскольку среднее арифметическое чувствительнее к выбросам, оно с большей вероятностью будет сдвигаться в большую или меньшую сторону при наличии небольшого числа экстремальных значений.

Задание: используя функцию median() посчитайте медианы для всех числовых переменных usedcars.

Обратимся к полученным медианным значениям в выводе результатов summary() для набора данных о подержанных автомобилях. Среднее арифметическое и медиана для цены довольно близки (различаются примерно на 5 %), однако разница между средним арифметическим и медианой пробега намного больше. Среднее арифметическое пробега составляет 44 261, что почти на 20 % больше медианы — 36 385. Поскольку среднее арифметическое более чувствительно к экстремальным значениям, чем медиана, тот факт, что среднее арифметическое получилось намного больше медианы, может навести на мысль, что в наборе данных есть подержанные автомобили с чрезвычайно высокими значениями пробега. Чтобы подробнее исследовать этот факт, нужно добавить в анализ дополнительную сводную статистику.

Измерение разброса: квартили и пятичисловая сводка

Среднее арифметическое и медиана позволяют быстро получить сводные значения, однако эти измерения средних значений мало скажут о том, существует ли разнообразие в измерениях. Для того чтобы измерить разнообразие, нужно использовать другой тип сводной статистики, касающийся разброса данных, — того, насколько тесно или широко распределены значения. Знание о разбросе дает представление о максимумах и минимумах данных, а также о том, насколько большинство значений близки к среднему арифметическому и медиане. Пятичисловая сводка — это набор из пяти статистических

показателей, которые приблизительно отображают разброс значений некоего признака. Все эти пять статистических показателей включены в выходные данные функции summary(). Перечислим их по порядку:

- минимум (Min.);
- первый квартиль, или Q1 (1st Qu.);
- медиана, или Q2 (Median); третий квартиль, или Q3 (3rd Qu.); максимум (Max.).

Как и следовало ожидать, минимальное и максимальное значения являются наиболее экстремальными значениями признака, указывая на наименьшее и наибольшее значения соответственно. Для вычисления этих значений для вектора в R есть функции min() и max().

Интервал между минимумом и максимумом называется диапазоном. В R есть функция range(), которая возвращает минимальное и максимальное значение:

```
> range(usedcars$price)
[1]  3800 21992
```

Сочетание range() и разностной функции diff() позволяет вычислить статистику диапазона в одной строке кода:

```
> diff(range(usedcars$price))
[1] 18192
```

Первый и третий квартили, Q1 и Q3, — это значения, ниже и выше которых расположена четверть всех значений. Наряду с медианой (Q2) квартили делят набор данных на четыре части, на каждую из которых приходится одинаковое количество значений. Квартили являются частным случаем квантилей, представляющих собой числа, которые делят данные на одинаковые по размеру множества. Кроме квартилей, в число часто используемых квантилей входят тертили (деление на три части), квинтили (пять частей), децили (десять частей) и процентили (сто частей).

Процентили часто используются для описания ранжирования значений. Например, студент, тестовый балл которого оценен на 99-й процентиль, показал результаты, которые как минимум не хуже, чем у 99 % других тестируемых.

Средние 50 % данных между первым и третьим квартилями представляют особый интерес, потому что это простая мера разброса. Разница между Q1 и Q3 называется межквартильным диапазоном (interquartile range, IQR) и может быть вычислена с помощью функции IQR():

```
> IQR(usedcars$price)
[1] 3909.5
```

Мы могли бы вычислить это значение для переменной usedcars$price вручную, по результатам summary(): 14 904 – 10 995 = 3909. Небольшая разница между нашими расчетами и результатом IQR() обусловлена тем, что R при выводе результатов summary() автоматически округляет числа.

Функция quantile() представляет собой универсальный инструмент для определения квантилей в наборе значений. По умолчанию функция quantile() возвращает пятичисловую сводку. Если применить эту функцию к переменной usedcars$price, то получим ту же сводную статистику, что и раньше:

```
> quantile(usedcars$price)
     0%     25%     50%     75%    100%
3800.0 10995.0 13591.5 14904.5 21992.0
```

Указав дополнительный параметр probs и используя вектор, содержащий точки разделения, можно получить произвольные квантили — например, 1-й и 99-й процентили:

```
> quantile(usedcars$price, probs = c(0.01, 0.99))
      1%      99%
5428.69 20505.00
```

Функция последовательности seq() генерирует векторы с равномерно распределенными значениями. Это облегчает получение других фрагментов данных, таких как квинтили, показанные в следующем примере:

```
> quantile(usedcars$price, seq(from=0, to=1, by=0.20))
     0%     20%     40%     60%     80%    100%
 3800.0 10759.4 12993.8 13992.0 14999.0 21992.0
```

Теперь, зная, что такое пятичисловая сводка, мы можем пересмотреть сводную статистику о подержанных машинах, полученную с помощью summary(). Что касается переменной price, то ее минимальное значение составляет 3800 долларов, а максимальное — 21 992. Интересно, что разница между минимумом и Q1 составляет около 7000 долларов, как и разница между Q3 и максимумом; однако разница между Q1 и медианой до Q3 составляет примерно 2000 долларов. Это говорит о том, что нижние и верхние 25 % значений разбросаны более широко, чем средние 50 % значений, которые, по-видимому, более тесно сгруппированы вокруг центра. Аналогичная тенденция наблюдается для переменной mileage. Данный паттерн разброса достаточно широко распространен, и его называют нормальным распределением данных. Разброс переменной mileage выявляет еще одно интересное свойство — разница между Q3 и максимумом намного больше, чем между минимумом и Q1. Другими словами, разброс между большими значениями пробега гораздо больше, чем между малыми. Это открытие помогает объяснить, почему среднее арифметическое оказалось намного больше, чем медиана. Поскольку среднее арифметическое чувствительно к экстремальным значениям, оно оказалось выше, в то время как медиана осталась на том же уровне. Это важное свойство, которое становится более очевидным, когда данные представлены визуально.

Задание: изучите пятичисловую сводку для переменных price и mileage.

Задание 5. Визуализация данных в R.

Изучите представленную теорию, выполните задания. Отчет о проделанной работе представьте в разделе отчета по практике «Язык R и среда разработки RStudio Cloud».

Визуализация числовых переменных: диаграммы размаха

Визуализация числовых переменных может быть полезна для выявления проблем с данными. Распространенным способом визуализации пятичисловой сводки является диаграмма размаха, также известная как «ящик с усами» (box-and-whisker). На диаграмме размаха отображаются центр и разброс числовой переменной в виде, позволяющем быстро получить представление о диапазоне и отклонении переменной или же сравнить ее с другими переменными. Рассмотрим диаграмму размаха для данных о цене и пробеге подержанных автомобилей. Чтобы получить диаграмму размаха для переменной, воспользуемся функцией boxplot(). Зададим пару дополнительных параметров, main и ylab, чтобы добавить к диаграмме заголовок и обозначить ось Y (вертикальную ось) соответственно. Команды построения диаграмм размаха для переменных price и mileage выглядят так:

```
> boxplot(usedcars$price, main = "Boxplot of Used Car Prices",
          ylab = "Price ($)")
> boxplot(usedcars$mileage, main = "Boxplot of Used Car Mileage",
          ylab = "Odometer (mi.)")
```

R выдаст следующие изображения (рис. 1).

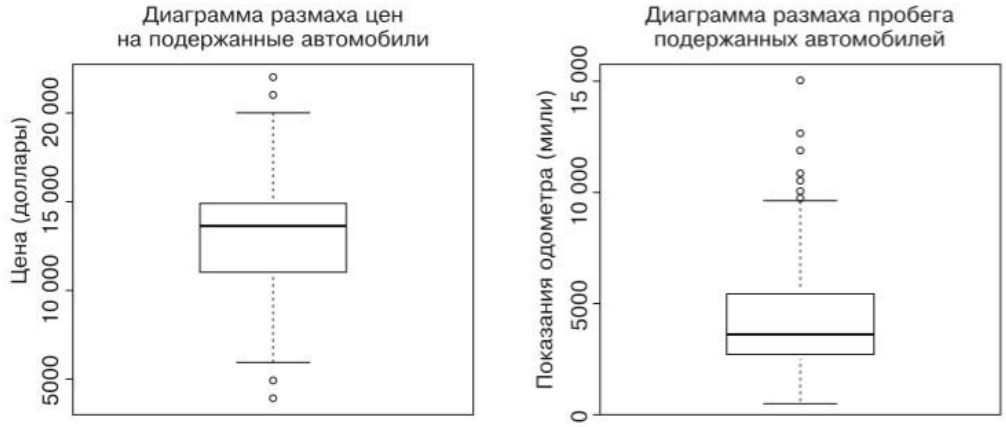


Рисунок 1. Диаграммы размаха для данных о цене и пробеге подержанных автомобилей

Диаграмма размаха отображает пятичисловую сводку, используя горизонтальные линии и точки. Горизонтальные линии, образующие прямоугольник в середине каждого рисунка, отражают значения Q1, Q2 (медиана) и Q3, если смотреть на диаграмму снизу вверх. Медиана обозначена темной линией, которая соответствует значениям 13 592 доллара по вертикальной оси для цены и 36 385 миль для пробега.

На простых диаграммах размаха, таких как диаграммы, показанные на рис. 1, ширина «ящика» является произвольной и не отображает какую-либо характеристику данных. Для более сложных видов анализа можно варьировать форму и размер «ящиков», чтобы облегчить сравнение данных по нескольким группам. Если вы хотите узнать больше о таких функциях, начните с изучения параметров notch и varwidth в документации к R-функции boxplot(), введя команду ?boxplot.

Минимальные и максимальные значения можно проиллюстрировать с помощью «усов», которые простираются ниже и выше «ящика»; однако распространенное соглашение допускает длину «усов» не более чем на 1,5 IQR ниже Q1 или выше Q3. Любые значения, которые выходят за пределы этого порога, считаются выбросами и обозначаются кружками либо точками. Так, напомню, что IQR для переменной price составляет 3909, притом что Q1 равно 10 995 и Q3 равно 14 904. Таким образом, выбросом является любое значение, меньшее чем 10 995 – 1,5 × 3909 = 5131,5 или большее чем 14 904 + 1,5 × 3909 = 20 767,5. На диаграмме размаха для переменной price видны два выброса сверху и снизу. На диаграмме размаха переменной mileage в нижней части выбросов нет, поэтому нижний «ус» простирается до минимального значения 4867. Вверху мы видим несколько выбросов, превышающих отметку 100 000 миль. Именно эти выбросы ответственны за то, что, как было видно, среднее арифметическое оказалось намного больше, чем медиана.

Визуализация числовых переменных: гистограммы

Гистограмма — это еще одно средство визуализации разброса значений числовой переменной. Гистограмма похожа на диаграмму размаха тем, что делит значения переменной на заранее определенное количество частей, или интервалов (bin), которые играют роль «контейнеров» для значений. Однако на этом сходство заканчивается. Если диаграмма размаха состоит из четырех частей, содержащих одинаковое количество значений, но различающихся по диапазону, то на гистограмме используется большее количество частей одинакового диапазона, а ее интервалы могут состоять из разного количества значений. Для того чтобы построить гистограмму для данных о цене и пробеге подержанных автомобилей, можно воспользоваться функцией hist(). Как и в случае с диаграммой размаха, зададим заголовок рисунка, используя параметр main, и введем имя оси X с помощью параметра xlab. Команды для построения гистограмм выглядят так:

```
> hist(usedcars$price, main = "Histogram of Used Car Prices",
       xlab = "Price ($)")
> hist(usedcars$mileage, main = "Histogram of Used Car Mileage",
       xlab = "Odometer (mi.)")
```

Гистограммы, которые будут созданы при этом, приведены на рис. 2.

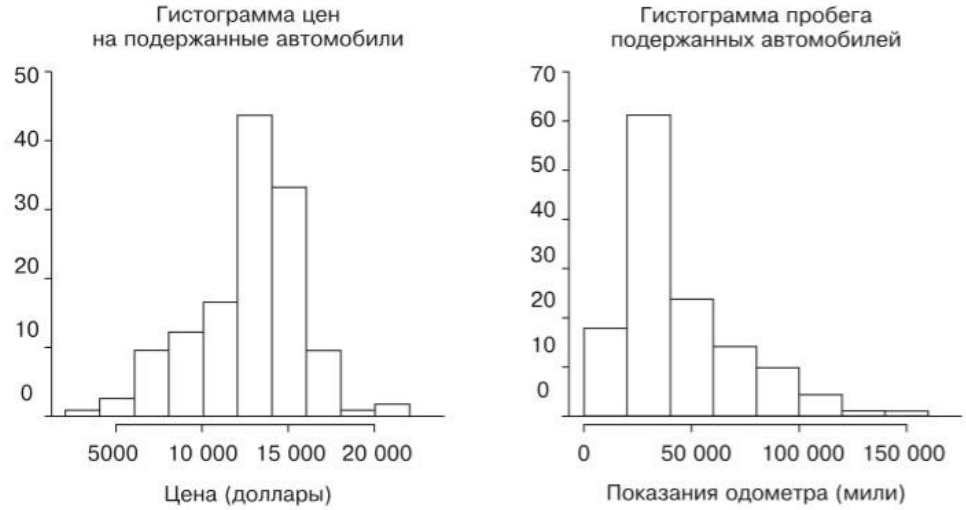


Рисунок 2. Гистограммы данных о цене и пробеге подержанных автомобилей

Гистограмма состоит из множества столбцов, высота которых соответствует количеству, или частоте, значений, попадающих в каждый интервал; все интервалы имеют равную ширину. Вертикальные линии, которые разделяют столбцы, как показано на горизонтальной оси, указывают на начальную и конечную точки диапазона значений, попадающих в данный интервал.

Возможно, вы заметили, что у приведенных выше гистограмм разное количество интервалов. Это связано с тем, что функция hist() пытается определить оптимальное количество интервалов для заданного диапазона переменной. Для того чтобы изменить значение, предлагаемое по умолчанию, используйте параметр breaks. Если присвоить этому параметру целочисленное значение, например breaks = 10, то будет создано ровно десять интервалов одинаковой ширины, а если задать вектор, такой как c(5000, 10000, 15000, 20000), то разделение на интервалы будет выполнено по указанным значениям.

На гистограмме переменной price каждый из десяти столбцов соответствует интервалу 2000 долларов, от 2000 до 22 000. Самый высокий столбец, расположенный в центре рисунка, соответствует диапазону от 12 000 до 14 000 долларов и частоте, равной 50. Поскольку известно, что данные охватывают 150 автомобилей, то треть всех автомобилей оценивается на сумму от 12 000 до 14 000 долларов. Почти 90 автомобилей — более половины — продаются по цене от 12 000 до 16 000 долларов.

Гистограмма переменной mileage состоит из восьми столбцов, представля ющих интервалы по 20 000 миль каждый, от 0 до 160 000 миль. В отличие от гистограммы переменной price, самый высокий столбец находится не в центре, а в левой части диаграммы; 70 автомобилей, попадающих в этот интервал, имеют показания одометра от 20 000 до 40 000 миль. Вы также могли заметить, что форма этих двух гистограмм несколько различна. Похоже, что цены на подержанные автомобили имеют тенденцию равномерно разделяться по обе стороны от середины, а пробеги автомобилей вытягиваются вправо. Эта характеристика известна как коэффициент асимметрии (skew) или, точнее, положительный коэффициент асимметрии, потому что верхние значения (правая сторона) разбросаны гораздо больше, чем нижние (левая сторона). Как показано на рис. 3, такие гистограммы данных выглядят растянутыми с одной стороны.

Рисунок 3. Три примера коэффициента асимметрии, показанные на идеализированных гистограммах

Возможность быстрой диагностики таких паттернов для наших данных является одной из сильных сторон гистограммы как инструмента исследования данных. Это станет еще важнее при изучении других моделей разброса числовых данных.

Интерпретация числовых данных: равномерное и нормальное распределение

Гистограммы, диаграммы размаха и статистические данные, описывающие средние значения и разброс, представляют собой способы исследования распределения значений переменной. Распределение переменной описывает вероятность попадания ее значения в различные диапазоны. Если все значения равновероятны — например, в наборе данных, в котором записаны значения, выпадающие при бросании игрального кубика, — то такое распределение называется равномерным. Равномерное распределение легко распознать на гистограмме, поскольку его столбцы примерно одинаковой высоты. Гистограмма равномерного распределения приведена на рис. 4. Важно отметить, что не все случайные события являются равномерно распределенными. Например, бросание шестигранного игрального кубика со смещенным центром тяжести приведет к тому, что одни числа будут выпадать чаще, чем другие. Каждый бросок кубика приводит к случайно выбранному числу, однако вероятность их выпадения неодинакова.

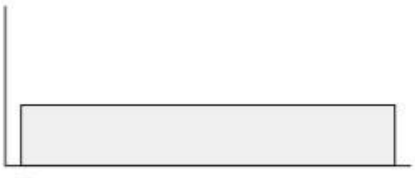


Рисунок 4. Равномерное распределение, представленное в виде идеализированной гистограммы

Возьмем, к примеру, данные о цене и пробеге подержанных автомобилей. Они явно неоднородны, поскольку одни значения, очевидно, более вероятны, чем другие. В сущности, на гистограмме переменной price видно, что значения становятся тем менее вероятными, чем дальше они находятся от обеих сторон центрального столбца, что приводит к колоколообразному распределению. Такое распределение настолько широко распространено среди реальных данных, что является отличительной чертой так называемого нормального распределения. Характерная колоколообразная кривая нормального распределения показана на рис. 5.

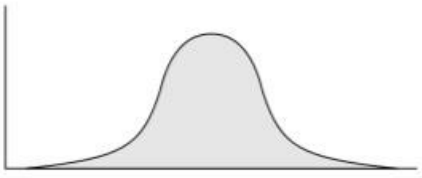


Рисунок 5. Нормальное распределение, показанное на идеализированной гистограмме

Существует множество типов распределений, не являющихся нормальными, однако многие явления реального мира создают данные, которые описываются с помощью

нормального распределения, поэтому свойства нормального распределения изучены очень подробно.

Измерение разброса: дисперсия и стандартное отклонение

Распределение позволяет характеризовать большое количество значений, используя меньшее количество параметров. Нормальное распределение, которое описывает множество типов реальных данных, может быть определено всего двумя параметрами: центром и разбросом. Центр нормального распределения определяется его средним арифметическим значением, которое было использовано ранее. Разброс измеряется статистическим параметром — стандартным отклонением.

Для того чтобы рассчитать стандартное отклонение, необходимо сначала вычислить дисперсию (variance, Var), которая определяется как среднеквадратическое отклонение между каждым значением и средним арифметическим. В математической нотации дисперсия набора из n значений в наборе с именем x определяется по формуле:

$$\mathrm{Var}(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2.$$

Греческая буква «мю» (внешне похожая на m или u) обозначает среднее арифметическое, а сама дисперсия обозначается греческой буквой «сигма» в квадрате. Стандартное отклонение (standard deviation, StdDev) представляет собой квадратный корень из дисперсии и обозначается буквой «сигма»:

$$\mathrm{StdDev}(X) = \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}.$$

В R для вычисления дисперсии и стандартного отклонения используются функции var() и sd(). Например, если вычислить дисперсию и стандартное отклонение по векторам price и mileage, то получим следующее:

```
> var(usedcars$price)
[1] 9749892
> sd(usedcars$price)
[1] 3122.482
> var(usedcars$mileage)
[1] 728033954
> sd(usedcars$mileage)
[1] 26982.1
```

При интерпретации дисперсии большие числа указывают на то, что данные разбросаны более широко относительно среднего значения. Стандартное отклонение показывает, насколько в среднем каждое значение отличается от среднего арифметического.

Если вычислить эти статистические данные вручную, используя формулы, приведенные выше, то результат будет несколько иной, чем если задействовать стандартные R-функции. Это связано с тем, что в приведенных формулах используется дисперсия совокупности (которая делится на n), а в R — выборочная дисперсия (которая делится на n – 1). За исключением очень маленьких наборов данных, различие невелико.

Стандартное отклонение может использоваться для того, чтобы быстро оценить, насколько экстремальным является данное значение, исходя из предположения, что распределение является нормальным. Правило 68–95–99,7 гласит, что при нормальном распределении 68 % значений находятся в пределах одного стандартного отклонения от среднего арифметического, в то время как 95 и 99,7 % значений находятся в пределах двух и трех стандартных отклонений соответственно. Это показано на рис. 6.

Рисунок 6. Процент значений в пределах одного, двух и трех стандартных отклонений от среднего арифметического при нормальном распределении

Применим эту информацию к данным о подержанных автомобилях. Извест но, что среднее арифметическое и стандартное отклонение цены составляли 12 962 и 3122 доллара соответственно. Таким образом, из предположения о нормальном распределении цен следует, что примерно 68 % автомобилей в приведенных данных рекламировались по ценам между 12 962 – 3122 = 9840 долларами и 12 962 + + 3122 = 16 804 долларами. Правило 68–95–99,7 применимо только в случае нормального распределения, однако базовый принцип работает для любых данных; значения, превышающие три стандартных отклонения от среднего арифметического, чрезвычайно редки.

Категориальные переменные

Как вы помните, набор данных о подержанных автомобилях содержит три категориальные переменные: model, color и transmission. В R эти переменные сохранены как символьные (chr) векторы, а не как тип factor, потому что при загрузке данных был использован параметр stringsAsFactors = FALSE. Кроме того, хотя переменная year хранится в виде числового (int) вектора, каждый год можно представить как категорию, применяемую к нескольким автомобилям. Поэтому переменную year также можно рассматривать как категориальную. По контрасту с числовыми данными категориальные данные обычно исследуются с использованием таблиц, а не сводной статистики. Таблица, представляющая одну категориальную переменную, называется таблицей частотности. Чтобы построить таблицу частотности для данных о подержанном автомобиле, можно воспользоваться функцией table():

```
> table(usedcars$year)
2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
   3    1    1    1    3    2    6   11   14   42   49   16    1
> table(usedcars$model)
SE   SEL   SES
78   23    49
> table(usedcars$color)
Black   Blue   Gold   Gray   Green   Red   Silver   White   Yellow
   35     17      1     16       5    25       32      16        3
```

Результатом работы table() являются списки категорий номинальной переменной и количество значений, попадающих в каждую категорию. Известно, что набор данных включает сведения о 150 подержанных автомобилях, и можно определить, что примерно треть всех автомобилей выпущена в 2010 году, учитывая, что 49 / 150 = 0,327. R также позволяет вычислить пропорции таблицы напрямую, с помощью команды prop.table() для таблицы, созданной функцией table():

```
> model_table <- table(usedcars$model)
> prop.table(model_table)
       SE          SEL          SES
0.5200000    0.1533333    0.3266667
```

Результаты prop.table() можно объединять с другими функциями R для представления вывода в другой форме. Предположим, что мы хотим отобразить результаты в процентах с одним десятичным знаком. Для этого можно умножить пропорции на 100, а затем воспользоваться функцией round(), указав digits = 1, как показано в следующем примере:

```
> color_table <- table(usedcars$color)
> color_pct <- prop.table(color_table) * 100
> round(color_pct, digits = 1)
Black   Blue   Gold   Gray   Green   Red   Silver   White   Yellow
 23.3   11.3    0.7   10.7     3.3  16.7     21.3    10.7      2.0
```

Это та же информация, что и в результате выполнения prop.table() по умолчанию, однако внесенные изменения облегчают чтение данных. Результаты показывают, что наиболее распространены черные машины, в этот цвет окрашена почти четверть (23,3 %) всех рекламируемых автомобилей. На втором месте находится серебристый с показателем 21,3 %, а на третьем — красный, 16,7 %. **Измерение средних значений: мода**

В статистической терминологии мода — это свойство, значение которого встречается чаще всего. Подобно среднему арифметическому и медиане, мода выступает еще одним показателем среднего значения. Обычно мода используется для категориальных данных, поскольку среднее арифметическое и медиана для номинальных переменных не определены.

Например, в данных о подержанных автомобилях мода переменной year равна 2010, для переменных model и color — SE и Black соответственно. Переменная может иметь более одной моды; переменная с одной модой является унимодальной, а переменная с двумя модами — бимодальной. Данные с несколькими модами называют мультимодальными.

Вы могли подумать, что для вычисления моды в R можно использовать функцию mode(), но в R эта функция используется не для вычисления статистической моды, а для получения типа переменной (числовой, списочный и т. п.). Чтобы получить статистическую моду, посмотрите на результат функции table() для категории с наибольшим количеством значений.

Одна или несколько мод используются для качественного анализа, позволяя понять важные значения. Однако опасно уделять много внимания модам, поскольку самое распространенное значение не всегда является доминирующим. Например, несмотря на то что черный был единственным самым популярным цветом автомобиля, такой цвет имеют не более четверти всех рекламируемых автомобилей. Лучше думать о соотношении мод с остальными категориями. Существует ли одна категория, которая доминирует над всеми остальными, или же таких категорий несколько? Подобные размышления о модах способны помочь генерировать проверяемые гипотезы, задавая вопросы о том, почему одни значения более распространены, чем другие. Если черный и серебристый — наиболее популярные цвета подержанных автомобилей, то можно предположить, что здесь описаны автомобили класса люкс, которые, как правило, выпускаются в более консервативных цветах. Или же эти цвета могут указывать на автомобили экономкласса, выбор цвета которых весьма ограничен. Запомним это и продолжим рассматривать данные. Считая моды стандартными значениями, мы можем применить концепцию статистической моды к числовым данным. Строго говоря, едва ли мода будет иметь смысл для непрерывной переменной, так как у нее не существует двух повторяющихся значений. Однако, если представить моду как самый высокий столбец на гистограмме, можно исследовать моды таких переменных, как price и mileage. Полезно будет изучить моду при исследовании числовых данных, в частности, чтобы проверить, являются ли данные мультимодальными (рис. 7).

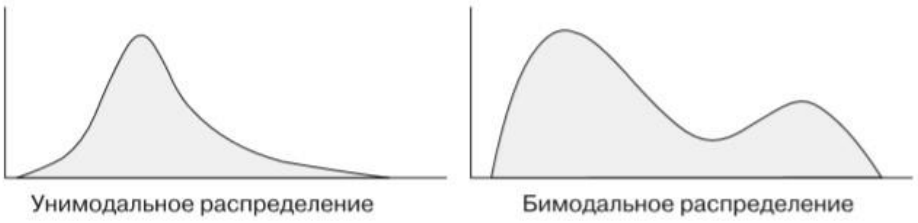


Рисунок 7. Гипотетические распределения числовых данных с одной и двумя модами

Взаимосвязи между переменными

До сих пор мы рассматривали переменные по отдельности, вычисляя только одномерную статистику. В процессе были заданы вопросы, на которые вы не могли ответить раньше.

- Означают ли данные о цене и пробеге, что исследуются только автомобили экономкласса, или среди данных есть автомобили класса люкс с большим пробегом? □□
- Можно ли на основании взаимосвязей между моделью и цветом сделать вывод о типе автомобилей, которые мы исследуем?

Ответы на подобные вопросы можно получить путем изучения двумерных отношений, то есть взаимосвязей между двумя переменными. Отношения между более чем двумя переменными называются многомерными. Начнем с двумерного случая.

Визуализация отношений: диаграммы разброса

Диаграмма разброса визуализирует двумерные отношения между числовыми признаками. Это двумерное изображение, где на координатной плоскости нанесены точки, так что значения одного признака откладываются по горизонтальной оси X, а значения другого признака — по вертикальной оси Y. Паттерны расположения точек отражают основные взаимосвязи между этими двумя признаками. Чтобы ответить на вопрос о соотношении цены и пробега, построим диаграмму разброса. Для этого воспользуемся функцией plot() с параметрами main, xlab и ylab. Чтобы применить plot(), нам нужно определить векторы x и y, содержащие значения, используемые для позиционирования точек на рисунке. Хотя выводы не зависят от того, какая именно переменная будет выбрана для оси X, а какая — для оси Y, общее соглашение таково, что в качестве переменной y выбирается та, которая предположительно зависит от другой (и поэтому называется зависимой переменной). Поскольку продавец не может изменить показания одометра автомобиля, пробег вряд ли будет зависеть от цены. Наоборот, гипотеза состоит в том, что цена автомобиля зависит от показаний одометра. Поэтому в качестве зависимой переменной y мы выберем цену. Полностью команда построения диаграммы разброса выглядит так:

```
> plot(x = usedcars$mileage, y = usedcars$price,
       main = "Scatterplot of Price vs. Mileage",
       xlab = "Used Car Odometer (mi.)",
       ylab = "Used Car Price ($)")
```

В результате получим диаграмму разброса, представленную на рис. 8. На данной диаграмме разброса четко видна зависимость между ценой на подержанный автомобиль и показаниями одометра. Чтобы прочитать диаграмму, изучите, как изменяются значения переменной y при увеличении значений на оси X. В этом случае, как правило, чем больше пробег, тем ниже цена автомобиля. Если вам когда-либо приходилось продавать или покупать подержанный автомобиль, то вряд ли это будет для вас открытием.

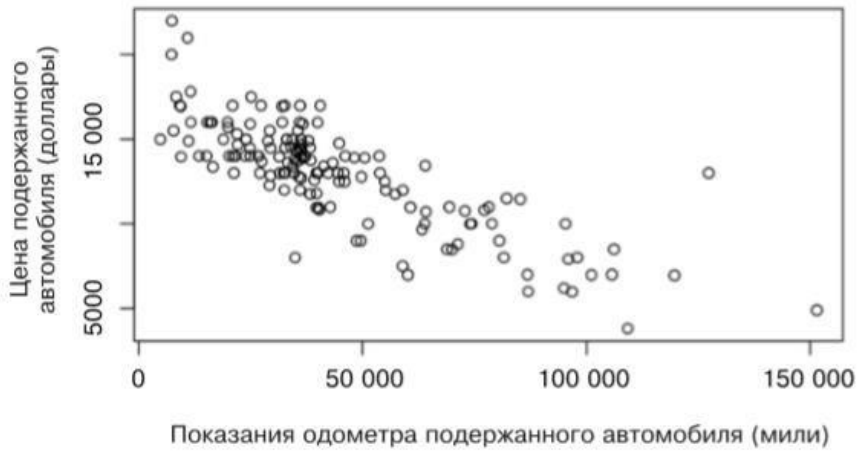


Рисунок 8. Зависимость цены подержанного автомобиля от пробега

Возможно, более интересен тот факт, что очень мало автомобилей имеют и высокую цену, и большой пробег, за исключением одинокого выброса на отметке 125 000 миль и 14 000 долларов. Отсутствие большего количества точек, подобных этой, свидетельствует о том, что этот набор данных вряд ли включает в себя автомобили класса люкс с большим пробегом. Все самые дорогие автомобили в наборе данных, особенно те, что стоят свыше 17 500 долларов, имеют очень низкий пробег. Это говорит о том, что мы имеем дело с автомобилями одного типа, которые, будучи новыми, продаются примерно за 20 000 долларов. Отношение, которое наблюдается, между ценами на автомобили и пробегом, — отрицательная связь, она образует паттерн точек, представляющий собой линию, направленную вниз. Аналогично положительная связь образует линию, направленную вверх. Ровная линия, или кажущийся случайным разброс точек, свидетельствует о том, что эти две переменные вообще не связаны. Сила линейной связи между двумя переменными измеряется величиной, известной как корреляция.
Имейте в виду, что не все связи образуют прямые линии. Иногда точки образуют U- или V-образную форму, а в других случаях паттерн значения переменных x или y кажется более или менее ярко выраженным. Такие модели подразумевают, что взаимосвязи между переменными не являются линейными.

Исследование взаимосвязей: перекрестные таблицы

Для изучения взаимосвязи между двумя номинальными переменными используется перекрестная таблица (также известная как кросс-таблица или таблица сопряженности). Кросс-таблица похожа на диаграмму разброса тем, что позволяет исследовать, каким образом значения одной переменной изменяются в зависимости от значений другой. В этой таблице строки представляют собой уровни одной переменной, а столбцы — уровни другой переменной. Число в каждой ячейке таблицы показывает количество значений, попадающих в конкретное пересечение строки и столбца. Чтобы ответить на вопрос о том, существует ли связь между моделью и цветом, рассмотрим кросс-таблицу.
В R есть несколько функций для создания перекрестных таблиц, включая table(), которую мы использовали для формирования таблиц частотности. Пожалуй, самой удобной является функция CrossTable() из пакета gmodels, созданного Грегори Р. Уорнсом (Gregory R. Warnes), так как она отображает в одной таблице проценты строк, столбцов и полей, избавляя нас от необходимости вычислять их самостоятельно. Чтобы установить пакет gmodels, введите следу ющую команду:

```
> install.packages("gmodels")
```

После установки пакета введите library(gmodels), чтобы загрузить пакет. Это необходимо делать в каждом сеансе R, в котором планируется использовать функцию CrossTable().
Прежде чем приступить к анализу, упростим наш проект, сократив количество уровней в переменной color. Эта переменная имеет девять уровней, но нам не нужно так много

подробностей. На самом деле нас интересует, относится ли цвет машины к консервативным. С этой целью мы разделим девять цветов на две группы. Первая группа будет включать в себя консервативные цвета: Black, Gray, Silver и White; вторая — Blue, Gold, Green, Red и Yellow. Создадим двоичную индикаторную переменную (часто называемую фиктивной переменной), указывающую, относится ли цвет машины к консервативным, согласно нашему определению. Эта переменная будет принимать значение 1, если цвет консервативный, и 0
— если нет:

```
> usedcars$conservative < usedcars$color %in% c("Black", "Gray", "Silver", "White")
```

Возможно, вы заметили новую команду: оператор %in% возвращает TRUE или FALSE для каждого значения вектора, стоящего с левой стороны от оператора, указывая, найдено ли в векторе значение, заданное с правой стороны. Проще говоря, эту строку можно истолковать так: «Подержанный автомобиль черного, серого, серебристого или белого цвета?».
Исследуя результат выполнения функции table() для нашей новой переменной, можно увидеть, что примерно две трети автомобилей имеют консервативные цвета, а одна треть — нет:

```
> table(usedcars$conservative)
FALSE  TRUE
51     99
```

Теперь посмотрим на кросс-таблицу, чтобы увидеть, как количество автомобилей консервативных цветов зависит от модели. Поскольку, предположительно, модель автомобиля диктует выбор цвета, рассмотрим индикатор консервативного цвета как зависимую переменную (y). Поэтому команда CrossTable() будет выглядеть так:

```
> CrossTable(x = usedcars$model, y = usedcars$conservative)
```

Результат будет следующим:

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  150

               | usedcars$conservative
usedcars$model |     FALSE |      TRUE | Row Total |
---------------|-----------|-----------|-----------|
            SE |        27 |        51 |        78 |
               |     0.009 |     0.004 |           |
               |     0.346 |     0.654 |     0.520 |
               |     0.529 |     0.515 |           |
               |     0.180 |     0.340 |           |
---------------|-----------|-----------|-----------|
           SEL |         7 |        16 |        23 |
               |     0.086 |     0.044 |           |
               |     0.304 |     0.696 |     0.153 |
               |     0.137 |     0.162 |           |
               |     0.047 |     0.107 |           |
---------------|-----------|-----------|-----------|
           SES |        17 |        32 |        49 |
               |     0.007 |     0.004 |           |
               |     0.347 |     0.653 |     0.327 |
               |     0.333 |     0.323 |           |
               |     0.113 |     0.213 |           |
---------------|-----------|-----------|-----------|
  Column Total |        51 |        99 |       150 |
               |     0.340 |     0.660 |           |
---------------|-----------|-----------|-----------|
```

В результатах функции CrossTable() содержится множество данных. Расположенная вверху легенда (с заголовком Cell Contents) указывает на то, как следует интерпретировать каждое значение. В строках таблицы указаны три модели подержанных автомобилей: SE, SEL и SES (и еще одна дополнительная строка для всех моделей). Столбцы указывают,

относится ли цвет машины к консервативным (плюс еще один столбец для обоих типов цветов). Первое значение в каждой ячейке указывает, какое количество автомобилей имеет данную комбинацию модели и цвета. Пропорции показывают вклад каждой ячейки в статистику хиквадрат, сумму строк, сумму столбцов и общую сумму таблицы. Наибольший интерес вызывает доля консервативных автомобилей для каждой модели. Пропорции строк говорят о том, что 0,654 (65 %) автомобилей модели SE окрашены в консервативные цвета, по сравнению с 0,696 (70 %) автомобилей модели SEL и 0,653 (65 %) модели SES. Эти различия относительно невелики, что свидетельствует об отсутствии существенных различий в типах цветов, выбранных для каждой модели автомобиля. Значения хи-квадрат означают вклад данной ячейки в критерий согласия Пирсона, или критерий согласия хи-квадрат для двух переменных. Данный критерий показывает, насколько вероятным является то, что разница в количестве ячеек в таблице обусловлена одной лишь случайностью. Если вероятность очень низкая, это дает убедительные доказательства того, что две переменные связаны. Для того чтобы вычислить критерий хи-квадрат, можно добавить в вызов функции CrossTable() дополнительный параметр: chisq = TRUE. В данном случае вероятность составляет около 93 %, а это говорит о том, что, весьма вероятно, различия в количестве ячеек обусловлены только случайностью, а не действительной взаимосвязью между моделью и цветом.

2. Список вопросов и (или) заданий для проведения промежуточной аттестации

Примеры индивидуальных заданий:

1. Разработайте предиктивную модель для классификации музыки по шести категориям.
2. Разработайте предиктивную модель для определения вероятности успеха заявки на получение гранта.
3. Разработайте предиктивную модель для определения вероятности того, что лекарственный препарат в процессе приема вызовет у пациента поражение печени.
4. Разработайте предиктивную модель, предсказывающую, откажется ли абонент от услуг страховой компании (интернет-провайдера/кабельного телевидения и пр.) на следующий месяц.
5. Разработайте предиктивную модель выявления рака.
6. Разработайте предиктивную модель предсказания кредитоспособности клиента банка.
7. Разработайте предиктивную модель предсказания стоимости медицинской страховки для конкретного клиента медицинской компании.
8. Разработайте предиктивную модель предсказания прочности бетона.
9. Разработайте предиктивную модель предсказания результатов тестирования по математике среди студентов первого курса.

Оценка в 100-балльной шкале	Оценка в 5-ти балльной шкале	Уровень сформированности компетенций
0-54 баллов	неудовлетворительно	недостаточный
55-69 баллов	удовлетворительно	базовый
70-85 баллов	хорошо	повышенный

86-100 баллов	отлично	

Приложение № 2 к рабочей программе дисциплины
«Технологическая (проектно-технологическая) практика»

Методические указания для студентов по освоению дисциплины

После окончания практики бакалавр представляет в трехдневный срок следующую отчетную документацию: дневник практики; отчет по результатам практики.

Требования к оформлению отчета.

Отчет выполняется на листах формата А4, текст печатается шрифтом Times New Roman, кегель 14 через 1,5 интервал.

Отчет по практике составляется каждым студентом индивидуально на основе материалов практики и индивидуального задания. Работа над отчетом должна вестись систематически в течение всего периода практики. Содержание излагается с соблюдением правил ЕСПД.

Структура отчета:

Титульный лист должен содержать: наименование учебного заведения и структурного подразделения (институт, кафедра), в котором обучается студент, шифр и наименование направления, название практики, место прохождения практики, ФИО студента, ФИО руководителя практики от кафедры, ФИО руководителя практики от предприятия, год прохождения практики.
Титульный лист подписывается автором, отчет проверяется и подписывается руководителями практики от предприятия и от кафедры.
Во введении определяются цели учебной практики и задачи для ее достижения. Примерный объем введения - 1-2 страницы.
В первом разделе «Язык R и среда разработки RStudio Cloud» приводится решение задач, направленных на изучение языка R и среды разработки RStudio Cloud.
Во втором разделе «Реализация индивидуального задания» приводится формулировка индивидуального задания. Далее выполняется описание математического аппарата, аппаратного и программного обеспечения ЭВМ, использованных для реализации поставленной задачи, непосредственная реализация решения поставленной задачи: поиск, подготовка и разметка данных; методы машинного обучения для выполнения индивидуального задания; метрики оценки результатов использования методов машинного обучения в рамках индивидуального задания; выбор инструментальных средств для выполнения индивидуального задания с обоснованием выбора; модели машинного обучения использованные при выполнении индивидуального задания.
В заключении на основе критического переосмысления накопленного опыта приводятся теоретические и практические выводы, результаты работы, дается оценка результатов собственной работы. Они должны излагаться в краткой форме и давать представление о степени выполнения задачи, поставленной перед студентом. Примерный объем заключения – 1 страница.

В списке литературы приводятся все источники, включая ресурсы сети интернет, использованные студентом в ходе прохождения производственной практики.