

2 Искусственный интеллект 3 в оценочной деятельности

4 Практическое руководство по разработке систем поддержки
5 принятия решений оценщиками с использованием языков
6 программирования R и Python

7 К. А. Мурашев

8 1 сентября 2021 г.

УДК 519(2+8+682)+004.891.2+330.4+338.5

ББК 16.6+22(16+17)+65.25

ГРНТИ 27.43.51+28.23.35+28.23.29+28.23.37+83.03.51

М91

Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python / К. А. Мурашев — Inkeri, Санкт-Петербург, 12 августа 2021 г. – 1 сентября 2021 г., 48 с.

Данное произведение является результатом интеллектуальной деятельности и объектом авторского права. Распространяется на условиях лицензии [Creative Commons Attribution-Share Alike 4.0 International \(CC BY-SA 4.0\)](#), оригинальный текст которой доступен по [ссылке](#) [5], перевод которого на русский язык доступен по [ссылке](#) [6]. Разрешается копировать, распространять, воспроизводить, исполнять, перерабатывать, исправлять и развивать произведение либо любую его часть в том числе и в коммерческих целях при условии указания авторства и лицензирования производных работ на аналогичных условиях. Все новые произведения, основанные на произведении, распространяемом на условиях данной лицензии, должны распространяться на условиях аналогичной лицензии, следовательно все производные произведения также будет разрешено распространять, изменять, а также использовать любым образом, в т. ч. и в коммерческих целях.

Программный код, разработанный автором и использованный для решения задач, описанных в данном произведении, распространяется на условиях лицензии [Apache License Version 2.0](#) [3], оригинальный текст которой доступен по [ссылке](#) [13], перевод текста которой на русский язык доступен по [ссылке](#) [3]. Программный код на языке R [63], разработанный автором, а также иные рабочие материалы к нему доступны по [ссылке](#) на портале Github [44], а также по [запасной ссылке](#) [45]. Программный код на языке Python [14], разработанный автором, а также иные рабочие материалы к нему доступны по [ссылке](#) на портале Github [46], а также по [запасной ссылке](#) [47].

В процессе разработки данного материала равно как и программного кода автор использовал операционную систему [Kubuntu](#) [9]. Для подготовки данного материала использовался язык \TeX [60] с набором макрорасширений $\text{\LaTeX} 2_{\epsilon}$ [61]. Конкретная техническая реализация заключается в использовании дистрибутива [TexLive](#) [62], редактора \LaTeX [40], компилятора \pdfLaTeX и системы цитирования \BibLaTeX / \Biber . Исходный код и дополнительные файлы, необходимые для его компиляции, доступны по [ссылке](#) на портале Github [49], а также по [запасной ссылке](#) [50].

Материал подготовлен в форме гипертекста: ссылки на ресурсы, размещённые в информационно-телекоммуникационной сети «Интернет» [105], выделены синим (blue) цветом, внутренние перекрёстные ссылки выделены красным (red) цветом, библиографические ссылки выделены зелёным (green) цветом. При подготовке данного материала использовался шаблон [KOMAScript Book](#) [34]. В целях облегчения понимания согласования слов в сложноподчинённых предложениях либо их последовательности в тексте реализована графическая разметка, позволяющая понять

структуру предложения: слова, согласованные между собой внутри предложения, подчёркнуты одинаковыми линиями, данное решение применяется только в тех предложениях, в которых, по мнению автора, возможно неоднозначное толкование в части согласования слов внутри него.

Данный материал выпускается в соответствии с философией *Rolling Release* [74], что означает что он будет непрерывно дорабатываться по мере обнаружения ошибок и неточностей, а также в целях улучшения внешнего вида. Идентификатором, предназначенным для определения версии материала, служат её номер и дата релиза, указанные на титульном листе, а также в колонтитулах. История версий приводится в таблице 0.1 на следующей странице-4. Актуальная версия перевода в формате PDF доступна по [ссылке](#) [49], а также по [запасной ссылке](#) [50].

В целях соответствия принципам [устойчивого развития](#) [30, 77], установленным в частности Стратегией [The European Green Deal](#) [53] и являющимся приоритетными для [Единой Европы](#) [24, 11, 68], а также содействия достижению [углеродной нейтральности](#) [64] рекомендуется использовать материал исключительно в электронной форме без распечатывания на бумаге.

Для связи с автором данного перевода можно использовать

- любой клиент, совместимый с протоколом [Tox](#) [54, 78], Tox ID = 2E71 CA29 AF96 DEF6 ABC0 55BA 4314 BCB4 072A 60EC C2B1 0299 04F8 5B26 6673 C31D 8C90 7E19 3B35;
- адрес электронной почты: kirill.murashev@tutanota.de;
- <https://www.facebook.com/murashev.kirill/> [1];

Реквизиты для оказания помощи проекту.

Тинькоф: +79219597644

BTC: [bc1qjzwtk3hc7ft9cf2a3u77cxflgnw93jktyjfs1?time=1627474534&exp=86400](#)

ETH:

Monero: 45ho 6Na3 dzoW DwYp 4ebD BXBr 6CuC F9L5 NGCD cсpa w2W4 W15a fiMM dGmf dhnp e6hP JSXk 9Mwm o9Up kh3a ek96 LFEa BZYX zGQ

USDT: 0x885e0b0E0bDCFE48750Be534f284EFfbEf6d247C

EURT: 0x885e0b0E0bDCFE48750Be534f284EFfbEf6d247C

CNHT: 0x885e0b0E0bDCFE48750Be534f284EFfbEf6d247C

История версий

Таблица 0.0.1: История версий материала

№	Номер версии	Дата	Автор	Описание
0	1	2	3	4
1	0.0001.0001	2021-08-14	KAM	Initial

Оглавление

86	1. Предисловие	17
87	2. Технологическая основа	25
88	2.1. Параметры использованного оборудования и программного обеспечения	25
89	2.2. Обоснование выбора языков R и Python в качестве средства анализа	
90	данных	25
91	2.2.1. Обоснование отказа от использования табличных процессоров	
92	в качестве средства анализа данных	25
93	2.2.2. R или Python	27
94	2.2.2.1. Общие моменты	27
95	2.2.2.2. Современное состояние	29
96	2.3. Система контроля версий Git	30
97	2.3.1. Общие сведения	30
98	2.3.2. Хеш-функции	33
99	2.3.3. Начало работы с Git и основные команды	34
100	2.3.4. Исключение файлов из списка отслеживания	47
101	2.3.5. Ветки проекта, указатели	47
102	2.3.6. Работа с ГитХаб	47
103	2.3.7. Rebase	47
104	2.3.8. Работа с Git в IDE	47
105	2.4. Установка и настройка	47
106	2.4.1. Git	47
107	2.4.1.1. Установка на операционных системах, основанных на De-	
108	bian: Debian, Ubuntu, Mint и т. п.	47
109	2.4.1.2. Установка на операционной системе Windows	48
110	2.4.1.3. Установка на macOS	48

¹¹¹ List of Algorithms

Рабочая версия

112 Список иллюстраций

113	2.3.1.Локальная система контроля версий	31
114	2.3.2.Схема работы централизованной системы контроля версий	32

115 Список таблиц

116	0.0.1 История версий материала	4
117	2.1.1.Параметры использованного оборудования	25
118	2.1.2.Параметры использованного программного обеспечения	26

Список литературы

- [1] URL: <https://www.facebook.com/murashev.kirill/> (дата обр. 28.07.2021).
- [2] Royal Institution Surveyors of Chartered (RICS). *RICS Valuation — Global Standards*. English. UK, London: RICS, 28 нояб. 2019. URL: <https://www.rics.org/eu/upholding-professional-standards/sector-standards/valuation/red-book/red-book-global/> (дата обр. 10.06.2020).
- [3] *Apache 2.0*. URL: http://licenseit.ru/wiki/index.php/Apache_License_version_2.0#.D0.A2.D0.B5.D0.BA.D1.81.D1.82_.D0.BB.D0.B8.D1.86.D0.B5.D0.BD.D0.B7.D0.B8.D0.B8 (дата обр. 17.08.2021).
- [4] Scott Chacon. *Pro Git book*. Перевод на русский язык. URL: <https://git-scm.com/book/ru/v2> (дата обр. 25.08.2021).
- [5] Creative Commons. *Creative Commons Attribution-ShareAlike 4.0 International*. нояб. 2013. URL: <https://creativecommons.org/licenses/by-sa/4.0/legalcode>.
- [6] Creative Commons. *Creative Commons Attribution-ShareAlike 4.0 International RUS*. нояб. 2013. URL: <https://creativecommons.org/licenses/by-sa/4.0/legalcode.ru>.
- [7] Microsoft Corporation. *Microsoft Excel*. Английский. URL: <https://www.microsoft.com/en-us/microsoft-365/excel> (дата обр. 20.08.2021).
- [8] CorVVin. *Хеш-функция, что это такое?* URL: <https://habr.com/en/post/534596/> (дата обр. 25.08.2021).
- [9] Kubuntu devs. *Kubuntu official site*. Kubuntu devs. URL: <https://kubuntu.org/> (дата обр. 17.08.2021).
- [10] KDE e.V. *Plasma. KDE community*. Английский. KDE e.V. URL: <https://kde.org/plasma-desktop/> (дата обр. 19.08.2021).
- [11] Institute Greater for a Europe. *Institute for a Greater Europe official site*. URL: <https://www.institutegreatereurope.com/> (дата обр. 15.04.2021).
- [12] StatSoft Europe. *Statistica: official site*. URL: <https://www.statistica.com/en/> (дата обр. 24.08.2021).
- [13] Apache Software Foundation. *Apache LicenseVersion 2.0*. Английский. URL: <https://www.apache.org/licenses/LICENSE-2.0> (дата обр. 17.08.2021).

- [14] Python Software Foundation. Английский. Python Software Foundation. URL: <https://www.python.org/> (дата обр. 17.08.2021).
- [15] The Apache Software Foundation. *OpenOffice Calc*. URL: <https://www.openoffice.org/product/calc.html> (дата обр. 20.08.2021).
- [16] The Document Foundation. *LibreOffice Calc*. Английский. URL: <https://www.libreoffice.org/discover/calc/> (дата обр. 20.08.2021).
- [17] The IFRS Foundation. *IFRS 13 Fair Value Measurement*. UK, London: The IFRS Foundation, 31 янв. 2016. URL: <http://eifrs.ifrs.org/eifrs/bnstandards/en/IFRS13.pdf> (дата обр. 10.06.2020).
- [18] *GeoGebra official site*. URL: <https://www.geogebra.org/> (дата обр. 26.08.2021).
- [19] *Git Download for Windows*. URL: <https://git-scm.com/download/win> (дата обр. 29.08.2021).
- [20] *Git install on macOS*. URL: <https://git-scm.com/download/mac> (дата обр. 29.08.2021).
- [21] *Git official site*. URL: <https://git-scm.com/> (дата обр. 19.08.2021).
- [22] *GitHub Desktop*. URL: <https://desktop.github.com/> (дата обр. 19.08.2021).
- [23] Google. *Google Sheets*. URL: <https://www.google.com/sheets/about/> (дата обр. 20.08.2021).
- [24] Lisbon-Vladivostok Work group. *Initiative Lisbon-Vladivostok*. URL: <https://lisbon-vladivostok.pro/> (дата обр. 15.04.2021).
- [25] *Homebrew*. URL: <https://brew.sh/> (дата обр. 29.08.2021).
- [26] IBM. *SPSS: official page*. URL: <https://www.ibm.com/products/spss-statistics> (дата обр. 24.08.2021).
- [27] IHS Global Inc. *Eviews: official site*. URL: <https://www.eviews.com/home.html> (дата обр. 24.08.2021).
- [28] SAS Institute Inc. *SAS: official site*. URL: https://www.sas.com/en_us/home.html (дата обр. 24.08.2021).
- [29] Intel. *Процессор Intel® Core™ i7-7500U*. Русский. тех. отч. URL: <https://ark.intel.com/content/www/ru/ru/ark/products/95451/intel-core-i7-7500u-processor-4m-cache-up-to-3-50-ghz.html> (дата обр. 19.08.2021).
- [30] Investopedia. *Sustainability*. URL: <https://www.investopedia.com/terms/s/sustainability.asp> (дата обр. 15.04.2021).
- [31] ISO. *Office Open XML*. URL: https://standards.iso.org/ittf/PubliclyAvailableStandards/c071692_ISO_IEC_29500-4_2016.zip (дата обр. 20.08.2021).
- [32] ISO/IEC. *ISO/IEC 10746-2:2009. Information technology — Open distributed processing — Reference model: Foundations — Part 2*. English. под ред. ISO/IEC. Standard. ISO/IEC, 15 дек. 2009. URL: <http://docs.cntd.ru/document/431871894> (дата обр. 01.03.2021).

- [33] ISO/IEC. *ISO/IEC 2382:2015. Information technology — Vocabulary*. English. под ред. ISO/IEC. ISO/EIC, 2015. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en> (дата обр. 01.03.2021).
- [34] Markus Kohm. *koma-script – A bundle of versatile classes and packages*. 1994–2020. URL: <https://ctan.org/pkg/koma-script> (дата обр. 28.01.2021).
- [35] *LaTeXDraw official page*. URL: <http://latexdraw.sourceforge.net/> (дата обр. 26.08.2021).
- [36] Licenseit.ru. *GNU General Public License*. URL: http://licenseit.ru/wiki/index.php/GNU_General_Public_License (дата обр. 23.08.2021).
- [37] Licenseit.ru. *GNU General Public License version 2*. URL: http://licenseit.ru/wiki/index.php/GNU_General_Public_License_version_2 (дата обр. 23.08.2021).
- [38] Licenseit.ru. *Python License version 2.1*. URL: http://licenseit.ru/wiki/index.php/Python_License_version_2.1 (дата обр. 23.08.2021).
- [39] StataCorp LLC. *Stata: official site*. URL: <https://www.stata.com/> (дата обр. 24.08.2021).
- [40] *LyX official site*. URL: <https://www.lyx.org/> (дата обр. 28.01.2021).
- [41] Machinelearning.ru. *Нормальное распределение*. URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5 (дата обр. 02.03.2021).
- [42] Machinelearning.ru. *Параметрические статистические тесты*. URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%8B (дата обр. 02.03.2021).
- [43] LLC Minitab. *Minitab: official site*. URL: <https://www.minitab.com/en-us/> (дата обр. 24.08.2021).
- [44] Kirill A. Murashev. R. URL: https://github.com/Kirill-Murashev/AI_for_valuers_R_source.
- [45] Kirill A. Murashev. R. URL: <https://web.tresorit.com/l/1Zgvt#kBA5FiY0Qtverp8Rjz6gyg>.
- [46] Kirill A. Murashev. R. URL: https://github.com/Kirill-Murashev/AI_for_valuers_Python_source.
- [47] Kirill A. Murashev. R. URL: <https://web.tresorit.com/l/VGZE5#XqySAkmjYODAIcOp1ZWpmg>.
- [48] Kirill A. Murashev. *RICS Valuation — Global Standards 2020. Russian translation*. TeX. 28 июля 2021. URL: <https://web.tresorit.com/l/oFpJF#xr3UGoxLvsszn4vAaHtjqw>.

- [49] Kirill A. Murashev. *Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python*. Inkeri. URL: https://github.com/Kirill-Murashev/AI_for_valuers_book.
- [50] Kirill A. Murashev. *Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python*. Inkeri. URL: https://web.tresorit.com/l/3xiTP#1p8pFnG_9No9izLFd09xaA.
- [51] *Notepad++ site*. URL: <https://notepad-plus-plus.org/> (дата обр. 29.08.2021).
- [52] Linux Kernel Organization. *The Linux Kernel Archives*. Linux Kernel Organization. URL: <https://www.kernel.org/> (дата обр. 26.08.2021).
- [53] European Parliament. *The European Green Deal*. 15 янв. 2020. URL: https://www.europarl.europa.eu/doceo/document/TA-9-2020-0005_EN.html (дата обр. 15.04.2021).
- [54] Tox Project. *Tox project official site*. URL: <https://tox.chat/> (дата обр. 09.03.2021).
- [55] *Qt*. Английский. URL: <https://www.qt.io/> (дата обр. 19.08.2021).
- [56] R Foundation. *The Comprehensive R Archive Network*. URL: <https://cran.r-project.org/> (дата обр. 24.08.2021).
- [57] *SHA3-512 online hash function*. URL: https://emn178.github.io/online-tools/sha3_512.html (дата обр. 25.08.2021).
- [58] Statsoft. *Solving trees*. URL: <http://statsoft.ru/home/textbook/modules/stclatre.html> (дата обр. 20.08.2021).
- [59] PBC Studio. *RStudio official site*. Английский. URL: <https://www.rstudio.com/> (дата обр. 19.08.2021).
- [60] CTAN team. *TeX official site*. English. CTAN Team. URL: <https://www.ctan.org/> (дата обр. 15.11.2020).
- [61] LaTeX team. *LaTeX official site*. English. URL: <https://www.latex-project.org/> (дата обр. 15.11.2020).
- [62] *TeXLive official site*. URL: <https://www.tug.org/texlive/> (дата обр. 15.11.2020).
- [63] The R Foundation. *The R Project for Statistical Computing*. Английский. The R Foundation. URL: <https://www.r-project.org/> (дата обр. 17.08.2021).
- [64] Wikipedia. *Carbon neutrality*. URL: https://en.wikipedia.org/wiki/Carbon_neutrality (дата обр. 15.04.2021).
- [65] Wikipedia. *COVID-19 pandemic*. Английский. URL: https://en.wikipedia.org/wiki/COVID-19_pandemic (дата обр. 18.08.2021).
- [66] Wikipedia. *Efficient-market hypothesis*. URL: https://en.wikipedia.org/wiki/Efficient-market_hypothesis (дата обр. 29.10.2020).

- [67] Wikipedia. *Euclidean distance*. URL: https://en.wikipedia.org/wiki/Euclidean_distance (дата обр. 18.08.2021).
- [68] Wikipedia. *Greater Europe*. URL: https://en.wikipedia.org/wiki/Greater_Europe (дата обр. 15.04.2021).
- [69] Wikipedia. *Kelly Johnson (engineer)*. URL: [https://en.wikipedia.org/wiki/Kelly%5C_Johnson_\(engineer\)](https://en.wikipedia.org/wiki/Kelly%5C_Johnson_(engineer)) (дата обр. 06.11.2020).
- [70] Wikipedia. *KISS principle*. URL: https://en.wikipedia.org/wiki/KISS_principle (дата обр. 06.11.2020).
- [71] Wikipedia. *List of Linux distributions : Debian — based*. URL: https://en.wikipedia.org/wiki/Category:Debian-based_distributions (дата обр. 26.08.2021).
- [72] Wikipedia. *Office Open XML*. URL: https://ru.wikipedia.org/wiki/Office_Open_XML (дата обр. 20.08.2021).
- [73] Wikipedia. *Robert Gentleman*. URL: [https://en.wikipedia.org/wiki/Robert_Gentleman_\(statistician\)](https://en.wikipedia.org/wiki/Robert_Gentleman_(statistician)) (дата обр. 25.08.2021).
- [74] Wikipedia. *Rolling Release*. URL: https://ru.wikipedia.org/wiki/Rolling_release (дата обр. 28.01.2021).
- [75] Wikipedia. *Ross Ihaka*. URL: https://en.wikipedia.org/wiki/Ross_Ihaka (дата обр. 25.08.2021).
- [76] Wikipedia. *SHA-3*. URL: <https://ru.wikipedia.org/wiki/SHA-3> (дата обр. 26.08.2021).
- [77] Wikipedia. *Sustainability*. English. URL: <https://en.wikipedia.org/wiki/Sustainability> (дата обр. 15.04.2021).
- [78] Wikipedia. *Wikipedia: Tox protocol*. URL: [https://en.wikipedia.org/wiki/Tox_\(protocol\)](https://en.wikipedia.org/wiki/Tox_(protocol)) (дата обр. 09.03.2021).
- [79] Wikipedia. *Архитектура компьютера*. Russian. URL: https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0 (дата обр. 06.08.2021).
- [80] Wikipedia. *Высокоуровневый язык программирования*. URL: https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F (дата обр. 23.08.2021).
- [81] Wikipedia. *Детерминированный алгоритм*. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC (дата обр. 25.08.2021).

- [82] Wikipedia. *Интегрированная среда разработки*. URL: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8 (дата обр. 29.08.2021).
- [83] Wikipedia. *Коллизия хеш-функции*. URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8 (дата обр. 25.08.2021).
- [84] Wikipedia. *Непараметрическая статистика*. URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0 (дата обр. 20.08.2021).
- [85] Wikipedia. *Переменная (математика)*. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0 (дата обр. 20.08.2021).
- [86] Wikipedia. *Переменная (программирование)*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_\(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\)](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (дата обр. 20.08.2021).
- [87] Wikipedia. *Полнота по Тьюрингу*. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83 (дата обр. 23.08.2021).
- [88] Wikipedia. *Принцип Дирихле*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5_\(%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0\)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5_(%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0)) (дата обр. 25.08.2021).
- [89] Wikipedia. *Расстояние городских кварталов*. URL: https://en.wikipedia.org/wiki/Taxicab_geometry (дата обр. 18.08.2021).
- [90] Wikipedia. *Сверхвысокоуровневый язык программирования*. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D1%85%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F (дата обр. 23.08.2021).
- [91] Wikipedia. *Свободная лицензия*. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F (дата обр. 23.08.2021).

- [92] Wikipedia. *Свободное программное обеспечение*. Русский. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 (дата обр. 18.08.2021).
- [93] Wikipedia. *Сильная форма Гипотезы эффективного рынка*. URL: https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D1%8B%D0%BD%D0%BA%D0%B0%D0%A2%D1%80%D0%B8_%D1%84%D0%BE%D1%80%D0%BC%D1%8B_%D1%80%D1%8B%D0%BD%D0%BE%D1%87%D0%BD%D0%BE%D0%B9_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8 (дата обр. 18.08.2021).
- [94] Wikipedia. *Сценарный язык*. URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA (дата обр. 23.08.2021).
- [95] Wikipedia. *Хеш-функция*. URL: <https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F> (дата обр. 25.08.2021).
- [96] *Xcode page*. URL: <https://developer.apple.com/xcode/> (дата обр. 29.08.2021).
- [97] Кирилл Кринкин. *Введение в архитектуру ЭВМ и элементы ОС. Курс лекций*. Русский. Computer Science Center. URL: <https://www.youtube.com/watch?v=FzN8zzMRTlw&list=PLlb7e2G7aSpRZ9wDzXI-VYpk59acLF0Ir> (дата обр. 23.08.2021).
- [98] связи и массовых коммуникаций Российской Федерации Министерство цифрового развития. *Свободное программное обеспечение в госорганах*. Русский. URL: <https://www.gnu.org/philosophy/free-sw.ru.html> (дата обр. 18.08.2021).
- [99] Фонд свободного программного обеспечения. *Что такое свободная программа?* Русский. Фонд свободного программного обеспечения. URL: <https://www.gnu.org/philosophy/free-sw.ru.html> (дата обр. 18.08.2021).
- [100] Программирование на C и C++. Онлайн справочник программиста на C и C++. *Оператор*. URL: <http://www.c-cpp.ru/books/operatory> (дата обр. 20.08.2021).
- [101] Виталий Радченко. *Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес*. URL: <https://habr.com/en/company/ods/blog/324402/> (дата обр. 20.08.2021).
- [102] Министерство финансов России. *Международный стандарт финансовой отчётности (IFRS) 13 «Оценка справедливой стоимости»*. с изменениями на 11 июля 2016 г. Russian. Russia, Moscow: Минфин России, 28 дек. 2015. URL: <https://normativ.kontur.ru/document?moduleId=1&documentId=326168#10> (дата обр. 10.06.2020).

- 381 [103] Министерство цифрового развития Российской Федерации. *Национальная*
382 *программа «Цифровая экономика Российской Федерации»*. 29 окт. 2020. URL:
383 <https://digital.gov.ru/ru/activity/directions/858/> (дата обр. 29.10.2020).
- 384 [104] Министерство экономического развития РФ. *Федеральные стандарты оцен-*
385 *ки*. URL: https://www.consultant.ru/document/cons_doc_LAW_126896/.
- 386 [105] Российская Федерация. *Федеральный Закон «Об информации, информацион-*
387 *ных технологиях и о защите информации»*. 149-ФЗ. Russian. Russia, Moscow,
388 14 июля 2006. URL: [https://normativ.kontur.ru/document?moduleId=1&](https://normativ.kontur.ru/document?moduleId=1&documentId=376603&cwi=22898)
389 [documentId=376603&cwi=22898](https://normativ.kontur.ru/document?moduleId=1&documentId=376603&cwi=22898) (дата обр. 07.07.2020).
- 390 [106] Российская Федерация. *Федеральный закон «Об оценочной деятельности в*
391 *Российской Федерации»*. 29 июля 1998. URL: [https://normativ.kontur.ru/](https://normativ.kontur.ru/document?moduleId=1&documentId=396506&cwi=7508)
392 [document?moduleId=1&documentId=396506&cwi=7508](https://normativ.kontur.ru/document?moduleId=1&documentId=396506&cwi=7508) (дата обр. 18.08.2021).

Глава 1.

Предисловие

«Лучший способ в чём-то
разобраться до конца — это
попробовать научить этому
компьютер».
Дональд Э. Кнут

Целью данной работы является попытка объединения наработок в областях оценочной деятельности и искусственного интеллекта. Автор предпринимает попытку доказать возможность применения современных технологий искусственного интеллекта в сфере оценки имущества, его эффективность и наличие ряда преимуществ относительно иных методов определения стоимости и анализа данных открытых рынков. В условиях заданного руководством России курса на цифровизацию экономики и, в особенности, на развитие технологий искусственного интеллекта [103] внедрение методов машинного обучения в повседневную практику оценщиков представляется логичным и необходимым.

Данная работа писалась в условиях распространения новой коронавирусной инфекции [65], внесшей дополнительный вклад в процессы цифровизации во всём мире. Можно по-разному относиться к проблематике данного явления, однако нельзя отрицать его влияние на общество и технологический уклад ближайшего будущего. Повсеместный переход на технологии искусственного интеллекта, замена человеческого труда машинным, беспрецедентный рост капитализации компаний, сделавших ставку на развитие интеллектуальной собственности, делают невозможным игнорирование необходимости цифровой трансформации оценочной деятельности в России.

Актуальность предложенного автором исследования заключается во-первых в том, что оно даёт практический инструментарий, позволяющий делать обоснованные, поддающиеся верификации выводы на основе использования исключительно объективных информации и данных,¹ непосредственно наблюдаемых на открытых рын-

¹По мнению автора, отличие между информацией и данными заключается в том, что под ин-

ках, без использования каких-либо иных их источников, подверженных субъективному влиянию со стороны их авторов. Во-вторых, предложенные и рассмотренные в данной работе методы обладают весьма широким функционалом, позволяющим использовать их при решении широкого круга задач, выходящих за рамки работы над конкретной оценкой. Важность обеих причин автор видит в том, что на 2021 год в России в сфере оценочной деятельности сложилась ситуация, которую можно охарактеризовать тремя состояниями:

- состояние неопределённости будущего отрасли;
- состояние интеллектуального тупика;
- состояние технологической отсталости.

Первая проблема заключается в неопределённости как правового регулирования отрасли, так и её экономики. Введённая около четырёх лет назад система квалификационных аттестатов оценщиков, на которую регулятор, заказчики и, возможно, часть самих оценщиков возлагали надежду как на фильтр, позволяющий оставить в отрасли только квалифицированных специалистов, сократить предложение оценочных услуг и, следовательно, способствовать росту вознаграждений за проведение оценки, не оправдала ожиданий. Несмотря на существенное сокращение

формацией понимаются:

- знания о предметах, фактах, идеях и т. д., которыми могут обмениваться люди в рамках конкретного контекста [32];
- знания относительно фактов, событий, вещей, идей и понятий, которые в определённом контексте имеют конкретный смысл [33],

таким образом, в контексте данного материала под информацией следует понимать совокупность сведений, образующих логическую схему: теоремы, научные законы, формулы, эмпирические принципы, алгоритмы, методы, законодательные и подзаконные акты и т. п.

Данные же представляют собой:

- формы представления информации, с которыми имеют дело информационные системы и их пользователи [32];
- поддающееся многократной интерпретации представление информации в формализованном виде, пригодном для передачи, связи или обработки [33],

таким образом, в контексте данного материала под данными следует понимать собой совокупность результатов наблюдений о свойствах тех или иных объектов и явлений, выраженных в объективной форме, предполагающей их многократные передачу и обработку.

Например: информацией является знание о том, что для обработки переменных выборки аналогов, имеющих распределение отличное от [нормального](#) [41], в общем случае, некорректно использовать [параметрические методы](#) [42] статистического анализа; данные в этом случае — это непосредственно сама выборка.

Иными словами, оперируя терминологией [архитектуры ЭВМ](#) [79], данные — набор значений переменных, информация — набор инструкций.

Во избежание двусмысленности в тексте данного материала эти термины приводятся именно в тех смыслах, которые описаны выше. В случае необходимости также используется более общий термин «сведения», обобщающий оба вышеуказанных понятия. В ряде случаев, термины используются в соответствии с принятым значением в контексте устоявшихся словосочетаний.

числа оценщиков, имеющих право подписывать отчёты об оценке, не произошло никаких значимых изменений ни в части объёма предложения услуг, ни в части уровня цен на них. Фактически произошло лишь дальнейшее развитие уже существовавшего ранее института подписантов отчётов — оценщиков, имеющих необходимые квалификационные документы и выпускающих от своего имени отчёты, в т. ч. и те, в подготовке которых они не принимали участия. В ряде случаев подписант мог и вовсе не читать отчёт либо даже не видеть его в силу своего присутствия в другом регионе, отличном от региона деятельности компании, выпустившей отчёт. При этом, как ни странно, доход таких «специалистов» не вырос существенным образом. Всё это очевидным образом приводит к недовольству регуляторов в адрес оценочного сообщества. В таких условиях следует ожидать неизбежного дальнейшего ужесточения регулирования и усугубления положения добросовестных оценщиков и оценочных компаний. Вместе с тем было бы ошибочным считать, что виной всему являются исключительно сами оценщики и их работодатели. В существенной степени проблемы квалификации и качества работы оценщиков вызваны не их нежеланием добросовестно выполнять свою работу, а отсутствием у заказчиков интереса к серьёзной качественной оценке. Не секрет, что в большинстве случаев оценка является услугой, навязанной требованиями закона либо кредитора, не нужной самому заказчику, которого очевидно волнует не качество отчёта об оценке, а соответствие определённой в нём стоимости ожиданиям и потребностям заказчика, его договорённостям с контрагентами. В таких условиях, с одной стороны, экономика не создаёт спрос на качественную оценку, с другой — сами оценщики не предлагают экономике интересные решения и новые ценности, которые могли бы принести в отрасль дополнительные финансовые потоки.

Вторая проблема тесно связана с первой и выражается в том числе в наблюдаемом на протяжении последних примерно 10 лет падении качества отчётов об оценке и общей примитивизации работы оценщика. Суть данной проблемы можно кратко сформулировать в одной фразе: «раньше молодые оценщики спрашивали „как проанализировать данные рынка и построить модель для оценки“, сейчас они задают вопрос „где взять корректировку на “X”“». Установление метода корректировок в качестве доминирующего во всех случаях даже без анализа применимости других методов стало логичным итогом процесса деградации качества отчётов об оценке. При этом источником подобных корректировок чаще всего являются отнюдь не данные открытого рынка. Как и в первом случае винить в этом только самих оценщиков было бы неправильным. В условиях работы в зачастую весьма жёстких временных рамках и за небольшое вознаграждение, оценщик часто лишён возможности провести самостоятельный анализ тех или иных свойств открытого рынка, вследствие и по причине чего вынужден использовать внешние нерыночные данные в том числе и непроверенного качества. Со временем это становится привычкой, убивающей творчество и стремление к поиску истины.

Третья проблема также неразрывно связана с двумя первыми. Отсутствие конкуренции, основанной на стремлении оказывать как можно более качественные услуги, недостаточная капитализация отрасли, выражающаяся в том числе в относительно невысоких зарплатах оценщиков, не вполне последовательное регули-

рование отрасли со стороны государства — всё это создаёт условия, при которых у оценщиков отсутствует стимул, а зачастую и возможность внедрять инновации.

Данная работа служит следующей основной цели: дать в руки оценщика инструменты, позволяющие ему просто и быстро извлекать полезные сведения из сырых данных открытых рынков, интерпретировать их, выдвигать гипотезы, выбирать среди них наиболее перспективные и в итоге получать готовые модели предсказания различных свойств объекта оценки, в том числе его стоимости. Есть некоторая надежда, что применение технологий искусственного интеллекта позволит, не увеличивая трудоёмкость, а скорее напротив, снижая её, повысить качество работы оценщика, усилить доказательную силу отчётов об оценке и в итоге позволит создать новые ценности, предлагаемые оценщиками экономике, государству, потребителям, а главное всему обществу.

Особенностью данной работы является её практическая направленность: в тексте содержатся все необходимые инструкции, формулы, описания и фрагменты программного кода либо ссылки на них, необходимые и достаточные для воспроизведения всех рассмотренных методов и их описания в отчётах об оценке.

Данная работа состоит из двух частей. Первая посвящена в большей степени теории, описанию методов, а также применению языка [R](#) [63]. Вторая имеет большую практическую направленность и содержит руководства по применению языка [Python](#) [14]. Объяснение данного факта содержится далее в разделе ССЫЛКА. В работе будут рассмотрены следующие вопросы:

- a) автоматизированный сбор данных с веб-ресурсов;
- b) семантический анализ текстов объявлений;
- c) работа с геоданными;
- d) первичная интерпретация и визуализация данных открытых рынков;
- e) проверка статистических гипотез;
- f) задачи классификации;
- g) корреляционный анализ;
- h) регрессионный анализ;
- i) анализ временных рядов;
- j) задачи многомерного шкалирования;
- k) байесовская статистика;
- l) деревья классификации;
- m) случайные леса;

- п) нейронные сети;
- о) глубокое обучение;
- р) обучение с подкреплением;
- q) нечёткая логика.

Вышеприведённый перечень не является исчерпывающим и будет дорабатываться по мере развития проекта.

Данная работа основана на четырёх основополагающих принципах и предпосылках.

- а) *Принцип «вся информация об активе учтена в его цене»*. Данный принцип говорит о том, что существует функциональная зависимость между ценой актива (обязательства) и его свойствами. Он тесно связан с [Гипотезой эффективного рынка \[66\]](#), лежащей в основе технического биржевого анализа. При этом для целей настоящей работы данная гипотеза принимается в её [сильной форме эффективности \[93\]](#). С точки зрения оценщика это означает, что нет необходимости искать какие-либо данные кроме тех, которые непосредственно и объективно наблюдаются на рынке.
- б) *Принцип «максимального использования релевантных наблюдаемых исходных данных и минимального использования ненаблюдаемых исходных данных»*. Данный принцип согласуется с требованиями п. 3 [Международного стандарта финансовой отчётности 13 «Оценка справедливой стоимости» \[102\]](#) (IFRS 13 [17]), а также, например, принципами [Всемирных стандартов оценки RICS \[48\]](#) (RICS Valuation — Global Standards [2]) и основывается на них. С точки зрения оценщика данный принцип означает, что лучшая практика оценки заключается в работе непосредственно с данными открытых рынков, а не чьей-либо их интерпретацией, существующей, например, в виде готовых наборов корректировок, порой весьма далёких от реальности.
- в) *Принцип KISS [70]* (keep it simple stupid, вариации: keep it short and simple, keep it simple and straightforward и т. п.), предложенный американским авиаинженером [Келли Джонсоном \[69\]](#), ставший официальным принципом проектирования и конструирования ВМС США с 1960 г. Данный принцип заключается в том, что при разработке той или иной системы следует использовать самое простое решение из возможных. Применительно к тематике данной работы это означает, что в тех случаях, когда автор сталкивался с проблемой выбора способа решения задачи в условиях неопределённости преимуществ и недостатков возможных вариантов, он всегда выбирал самый простой способ. Например в задаче кластеризации, выбирая между видами расстояний, автор делает выбор в пользу [евклидова](#) либо [манхэттенского](#) расстояний [67, 89].

d) *Принцип «не дай алгоритму уничтожить здравый смысл».* Данный принцип означает необходимость самостоятельного осмысления всех результатов выполнения процедур, в т. ч. и промежуточных. Возможны ситуации, когда полученные результаты могут противоречить здравому смыслу и априорным знаниям о предметной области, которыми обладает оценщик либо пользователи его работы. Следует избегать безоговорочного доверия к результатам, выдаваемым алгоритмами. Если построенная модель противоречит априорным знаниям об окружающей реальности, то следует помнить, что другой реальности у нас нет, тогда как модель может быть скорректирована либо заменена на другую.

Все описанные этапы действий описаны таким образом, что позволяют сразу же без каких-либо дополнительных исследований воспроизвести всё, что было реализовано в данной работе. От пользователей потребуется только установить необходимые программные средства, создать свой набор данных для анализа и загрузить его в пакет. Все действия по установке и настройке описаны внутри данного руководства. Важным аспектом является то обстоятельство, что при подготовке данного исследования использовалось исключительно [свободное программное обеспечение](#) [99, 92, 98]. Таким образом, любой читатель сможет воспроизвести все описанные действия без каких-либо затрат на приобретение тех или иных программных продуктов.

От пользователей данного руководства не требуется наличие специальных познаний в области разработки программного обеспечения, software engineering и иных аспектов computer science. Некоторые понятия вроде «класс», «метод», «функция», «оператор», «регулярные выражения» и т. п. термины из сферы программирования могут встречаться в тексте руководства, однако их понимание либо непонимание пользователем не оказывает существенного влияния на восприятие материала в целом. В отдельных случаях, когда понимание термина является существенным, как например в случае с термином «переменная», в тексте руководства приводится подробное объяснение смысла такого термина, доступное для понимания неспециалиста.

Также от пользователей руководства не требуется (хотя и является желательным) глубокое понимание математической статистики, дифференциальных вычислений, линейной алгебры, комбинаторики, методов исследования операций, методов оптимизации и иных разделов математики и математической статистики, хотя и предполагается наличие таких познаний на уровне материала, включённого в школьную программу и программу технических и экономических специальностей вузов России. В тексте руководства приводится описание смысла и техники всех применённых статистических методов, математических операций и вычислений в объёме, достаточном, по мнению автора, для обеспечения доказательности при использовании методов, рассмотренных в данной работе. Автор всегда приводит ссылки на материалы, подтверждающие приведённые им описания за исключением случаев общеизвестных либо очевидных сведений. Особое внимание автор уделяет соблюдению требований к информации и данным, имеющим существенное значение

для определения стоимости объекта оценки, установленных Федеральным законом «Об оценочной деятельности в Российской Федерации» [106], а также Федеральными стандартами оценки [104].

Сведения, приведённые в настоящем руководстве, являются, по мнению автора, достаточными для обеспечения выполнения вышеуказанных требований к информации, содержащейся в отчёте об оценке. Таким образом, использование описаний процедур, приведённых в настоящем руководстве, скорее всего должно быть достаточным при использовании изложенных в нём методик в целях осуществления оценочной деятельности и составления отчёта об оценке. Однако, автор рекомендует уточнять требования, предъявляемые к отчёту об оценке со стороны саморегулируемой организации, в которой состоит оценщик, а также со стороны заказчиков и регуляторов.

В силу свободного характера лицензии, на условиях которой распространяется данная работа, она, равно как и любая её часть, может быть скопирована, воспроизведена, переработана либо использована любым другим способом любым лицом в т. ч. и в коммерческих целях при условии распространения производных материалов на условиях такой же лицензии. Таким образом, автор рекомендует использовать тексты, приведённые в настоящем руководстве для описания выполненных оценщиком процедур.

По мнению автора, данное руководство и описанные в нём методы могут быть особенно полезны в следующих предметных областях:

- оценка и переоценка залогов и их портфелей;
- контроль за портфелями залогов со стороны регулятора банковской сферы;
- оценка объектов, подлежащих страхованию, и их портфелей со стороны страховщиков;
- оценка объектов со стороны лизинговых компаний;
- оценка больших групп активов внутри холдинговых компаний и предприятий крупного бизнеса;
- оценка в целях автоматизированного налогового контроля;
- государственная кадастровая оценка;
- экспертиза отчётов об оценке, контроль за деятельностью оценщиков со стороны СРО.

Иными словами, особая ценность применения методов искусственного интеллекта в оценке возникает там, где имеет место необходимость максимальной беспристрастности и незаинтересованности в конкретном значении стоимости.

В данном руководстве не содержатся общие выводы касательно параметров открытых рынков как таковых, не выводятся общие формулы, применимые всегда

и для всех объектов оценки. Вместо этого в распоряжение пользователей предоставляется набор мощных инструментов, достаточный для моделирования ценообразования на любом открытом рынке, определения стоимости любого объекта оценки на основе его актуальных данных. В случае необходимости пользователь, применяя рассмотренные методы, может самостоятельно разработать предсказательную модель для любых рынков и объектов. Забегая вперёд, можно сказать, что при решении конкретной практической задачи применение всех описанных методов не является обязательным, а если быть точным — явно избыточным. В тексте руководства содержатся рекомендации по выбору методов на основе имеющихся свойств данных, рассматриваются сильные и слабые стороны каждого из них.

Несмотря на изначально кажущуюся сложность и громоздкость методов, при более детальном знакомстве и погружении в проблематику становится ясно, что применение предложенных реализаций методов существенно сокращает время, необходимое для выполнения расчёта относительно других методов сопоставимого качества, а сама процедура сводится к написанию и сохранению нескольких строк кода при первом применении и их вторичному многократному использованию для новых наборов данных при будущих исследованиях.

Автор выражает надежду, что данное руководство станет для кого-то первым шагом на пути изучения языков [R](#) [63] и [Python](#) [14], а также погружения в мир анализа данных, искусственного интеллекта и машинного обучения.

Глава 2.

Технологическая основа

2.1. Параметры использованного оборудования и программного обеспечения

При выполнении всех описанных в данной работе процедур, равно как и написании её текста использовалась следующая конфигурация оборудования.

Таблица 2.1.1. Параметры использованного оборудования

№	Категория	Модель (характеристика)	Источник
0	1	2	3
1	Процессор	4 × { Intel ® Core ™ i7-7500U CPU @ 2.70GHz	[29]
2	Память	11741076B	

При выполнении всех описанных в данной работе процедур, равно как и написании её текста использовалась следующая конфигурация программного обеспечения.

Как видно из таблиц 2.1, 2.1 для анализа данных и разработки систем поддержки принятия решений на основе искусственного интеллекта вполне достаточно оборудования, обладающего средними характеристиками, а также свободных или, по крайней мере, бесплатных программных средств.

2.2. Обоснование выбора языков R и Python в качестве средства анализа данных

2.2.1. Обоснование отказа от использования табличных процессоров в качестве средства анализа данных

На сегодняшний день очевиден факт того, что доминирующим программным продуктом, используемым в качестве средства выполнения расчётов, в среде русских оценщиков является приложение MS Excel [7]. Следом за ним идут его бесплатные аналоги LibreOffice Calc и OpenOffice Calc [16, 15], первый из которых является

Таблица 2.1.2. Параметры использованного программного обеспечения

№	Категория/наименование	Значение/версия	Источник
0	1	2	3
1	Операционная система	Kubuntu 20.04	[9]
2	KDE Plasma	5.18.5	[10]
3	KDE Frameworks	5.68.0	[10]
4	Qt	5.12.8	[55]
5	R	4.1.1 (2021-08-10) "— "Kick Things"	[63]
6	RStudio	1.4.1717	[59]
7	Git	2.25.1	[21]
8	Github Desktop	2.6.3-linux1	[22]
9	Geogebra Classic	6.0.660.0-offline	[18]
10	LaTeXDraw	4.0.3-1	[35]
11	Python	3.8.10	
12	Spyder	3.3.6	
13	PyCharm Community	2021.2.1	
14	Kate	19.12.3	

также не только бесплатным, но и свободным программным обеспечением [99, 92, 98]. В ряде случаев используется Google Sheets [23]. Не оспаривая достоинства этих продуктов, нельзя не сказать о том, что они являются универсальными средствами обработки данных общего назначения и, как любые универсальные средства, сильны своей многофункциональностью и удобством, но не шириной и глубиной проработки всех функций. Во всех вышеуказанных программных продуктах в виде готовых функций реализованы некоторые основные математические и статистические процедуры. Также само собой присутствует возможность выполнения расчётов в виде формул, собираемых вручную из простейших операторов [100]. Однако возможности этих продуктов для профессионального анализа данных абсолютно недостаточны. Во-первых, в них имеются ограничения на размер и размерность исследуемых данных. Во-вторых, в них отсутствуют средства реализации многих современных методов анализа данных. Если первое ограничение не столь важно для оценщиков, редко имеющих дела с по-настоящему большими наборами данных и существенным числом переменных [85, 86] в них, второе всё же накладывает непреодолимые ограничения на пределы применимости таких программных продуктов. Например, ни одно из вышеперечисленных приложений не позволяет использовать методы непараметрической статистики [84] либо, например, решить задачи построения деревьев классификации [58] и их случайных лесов [101]. Таким образом, следует признать, что, оставаясь высококачественными универсальными средствами для базовых расчётов, вышеперечисленные приложения не могут быть использованы для профессионального анализа данных на современном уровне.

При этом их использование порой бывает необходимым на первоначальном исследовании. Некоторые исходные данные, предоставляемые оценщику для обработки,

содержатся в электронных таблицах. Такие таблицы помимо полезных сведений могут содержать посторонние данные, тексты, графики и изображения. В практике автора был случай предоставления ему для анализа данных в форме электронной таблицы формата [xlsx](#) [72, 31], имеющей размер около 143 МБ, содержащей помимо подлежащей анализу числовой информации о товарах их рекламные описания в текстовом виде и фотографии, составляющие свыше 90 % размера файла. Тем не менее просмотр исходных данных средствами табличных процессоров и создание нового файла, содержащего только необходимые для анализа данные, нередко является подготовительным этапом процесса анализа. В последующих разделах будут даны практические рекомендации касательно его реализации. По мнению автора, по состоянию на 2021 год лучшим табличным процессором является [LibreOffice Calc](#) [16], превосходящий [MS Excel](#) [7] по ряду характеристик.

2.2.2. R или Python

2.2.2.1. Общие моменты

Можно с уверенностью сказать, что по состоянию на второе полугодие 2021 года доминирующими и самыми массовыми техническими средствами анализа данных, машинного обучения и разработки искусственного интеллекта¹ являются языки программирования [R](#) [63] и [Python](#) [14]. Оба они являются [сверхвысокоуровневыми](#) [90] [сценарными](#) (скриптовыми) [94] языками программирования. Высокоуровневым называется такой язык программирования, в основу которого заложена сильная абстракция, т. е. свойство описывать данные и операции над ними таким образом, при котором разработчику не требуется глубокое понимание того, как именно машина их обрабатывает и исполняет [80]. [Сверхвысокоуровневым](#) [90] языком является такой язык программирования, в котором реализована очень сильная абстракция. Иными словами, в отличие от [языков программирования высокого уровня](#) [80], в коде, разработанном на [сверхвысокоуровневых языках](#) [90] описывается лишь принцип «как нужно сделать», код, выполненный на [сверхвысокоуровневых языках](#) [90] описывает лишь принцип «что нужно сделать». [Сценарным](#) (скриптовым) [94] языком называется такой язык программирования, работа которого основана на исполнении сценариев, т. е. программ, использующих уже готовые компоненты. Таким образом, можно сделать вывод, что [сверхвысокоуровневые языки](#) лучше всего подходят для тех, кто только начинает погружаться в программирование и не обладает экспертными знаниями в вопросах [архитектуры ЭВМ](#) [79].²

Оба языка распространяются на условиях [свободных лицензий](#) [91] с незначительными отличиями. [R](#) распространяется на условиях лицензии [GNU GPL 2](#) [37], [Python](#) — на условиях лицензии [Python Software Foundation License](#) [38], являющейся совместимой с [GNU GPL](#) [36]. Отличия между ними не имеют никакого практического значения для целей настоящего руководства и применения любо-

¹Разница между этими понятиями будет описана далее в ССЫЛКА

²Для первичного ознакомления с вопросами архитектуры ЭВМ автор рекомендует просмотреть [данный курс лекций](#) [97].

го из этих языков в оценочной деятельности в целом. Следует лишь знать основной факт: использование этих языков является легальным и бесплатным в том числе и для коммерческих целей. Основное отличие между этими языками заключается в частности в том, что Python — язык общего назначения, широко применяемый в различных областях, тогда как R — специализированный язык статистического анализа и машинного обучения. В целом можно сказать, что задачи анализа данных могут одинаково успешно решаться средствами обоих языков. Также они оба являются [Тьюринг-полными](#) [87] языками.

Преимущества R основаны на том факте, что он изначально был разработан двумя профессиональными статистиками: [Ross Ihaka](#) [75], [Robert Gentleman](#) [73], по первым буквам имён которых он и был назван. Дальнейшее развитие языка также осуществляется прежде всего силами профессиональных математиков и статистиков, вследствие чего для R реализовано значительное количество библиотек, выполняющих практически все доступные на сегодняшнем уровне развития науки статистические процедуры. Кроме того, можно быть уверенным в абсолютной корректности всех алгоритмов, реализованных в этих библиотеках. К тому же этот язык особенно популярен в академической среде, что означает факт того, что в случае, например, выхода какой-то статьи, описывающей новый статистический метод, можно быть уверенным, что соответствующая библиотека, реализующая этот метод выйдет в ближайшее время либо уже вышла. Кроме того, важным преимуществом R являются очень хорошо проработанные средства вывода графической интерпретации результатов анализа.

Недостатки R, как это часто бывает, следуют из его достоинств. Язык и его библиотеки поддерживаются в первую очередь силами математиков-статистиков, а не программистов, что приводит к тому, что язык относительно плохо оптимизирован с точки зрения software engineering, многие решения выглядят неочевидными и неоптимальными с точки зрения способов обращения к памяти, интерпретации в машинные команды, исполнения на процессоре. Это приводит к высокому потреблению ресурсов машины, в первую очередь памяти, медленному исполнению процедур. При этом, говоря о медленном исполнении, следует понимать относительность этой медлительности. Выполнение команды за 35 мс вместо 7 мс не замечается человеком и обычно не имеет сколько-нибудь определяющего значения. Проблемы с производительностью становятся заметны только при работе с данными большой размерности: миллионы наблюдений, тысячи переменных. В практических задачах, с которыми сталкиваются оценщики, подобная размерность данных выглядит неправдоподобной, вследствие чего можно говорить об отсутствии существенных недостатков языка R для целей применения в оценочной деятельности в целом и в целях задач, решаемых в данном руководстве, в частности. Следующей условной проблемой R является огромное количество библиотек³ и ещё более огромное количество возможных вариантов решения задач и предлагаемых для этого методов. Даже опытный аналитик может растеряться, узнав о том, что его задача может быть ре-

³По состоянию на 24 августа 2021 существует 18089 официальных библиотек, содержащихся на [официальной странице](#) [56] проекта.

шena десятками способов, выбор лучшего из которых сам по себе является нетривиальной задачей. Данную особенность конечно же нельзя считать недостатком самого языка R.

Преимуществом Python является его универсальность и существенно большая распространённость. Освоение основ данного языка для целей одной предметной области может быть полезным в дальнейшем, если по каким-то причинам оценщик захочет решать с его помощью задачи иного класса. Данный язык разработан и поддерживается профессиональными программистами, что означает его относительно приемлемую оптимизацию, превосходящую R, но уступающую, например C++.

К недостаткам Python можно отнести меньшее число библиотек, содержащих статистические процедуры. Кроме того, нет такой же уверенности в безупречности их алгоритмов. При этом следует отметить, что подобные риски присутствуют лишь в новых библиотеках, реализующих экспериментальные либо экзотические статистические процедуры. Для целей оценки как правило вполне достаточно уже относительно отработанных и проверенных библиотек.

Подводя итог, можно сказать, что нет однозначного ответа, какой из вышеупомянутых языков является предпочтительным для целей анализа данных в оценке. R развивается, оптимизируется и всё больше избавляется от «детских болезней» неоптимизированности, для Python создаются новые мощные библиотеки статистического анализа. Поэтому вопрос остаётся открытым.

Следует кратко упомянуть о том, что помимо R и Python в целях анализа данных также используются вендорские программные продукты такие как SAS [28], SPSS [26], Statistica [12], Minitab [43], Stata [39], EvIEWS [27] и ряд других. Однако все они являются платными, при этом стоимость лицензии на самый мощный из них — SAS начинается, как правило, от нескольких десятков тысяч долларов. В остальном, кроме привычного для большинства пользователей графического интерфейса они не имеют явных преимуществ перед R и Python, предоставляя при этом даже меньше возможностей.

2.2.2.2. Современное состояние

Вышеприведённый текст, содержащийся в предыдущей секции (2.2.2.1) был написан автором в 2019 году. За прошедший период произошли некоторые изменения, требующие внимания. В настоящее время Python серьёзно опережает R по распространённости в среде аналитиков данных. Можно говорить о некотором консенсусе, согласно которому R является средством разработки и анализа данных для научных целей, тогда как Python применяется в бизнес среде. Несмотря на это, автор считает, что в целях анализа данных данные языки вполне взаимозаменяемы. Некоторые библиотеки портированы из одного из них в другой. При этом нельзя не признать, что за последние годы R существенно сдал позиции в пользу Python. В особенности это справедливо именно для российского рынка разработки систем анализа данных. Определённый пик интереса к R в России имел место в 2015–2017 годах, после чего его популярность пошла на спад. В мире пик интереса к R пришёлся на 2016–2018 годы после чего его популярность стабилизировалась. Язык продолжает активно

815 развивается.

816 В российской практике коммерческого анализа данных его заказчики, как прави-
817 ло, требуют реализации на Python, применение вместо него R чаще всего приходит-
818 ся обосновывать отдельно. Таким образом, можно говорить о том, что применение
819 Python де факто является стандартом. Кроме того, продвижению Python во всём
820 мире способствует позиция компаний интернет-гигантов, использующих его в сво-
821 их системах машинного обучения. Следующим фактором успеха Python является
822 его широкое распространение в теме разработки нейронных сетей, также являющее-
823 ся следствием практик крупных IT-компаний. Также Python широко распространён
824 и за пределами области анализа данных, что означает существенно большее число
825 специалистов, владеющих им. При этом для R разработан ряд уникальных отрас-
826 левых библиотек, содержащих специфические функции. R безоговорочно лидирует
827 в области биоинформатики, моделирования химических процессов, социологии.

828 При этом, R по-прежнему предоставляет существенно более широкие возмож-
829 ности визуализации, а также позволяет легко разрабатывать веб-интерфейсы по-
830 средством [Shiny](#). R имеет отличный инструмент написания документации к коду
831 в процессе разработки самого кода — [R Markdown](#).

832 Подводя итоги, можно сказать о том, что современным оценщикам следует иметь
833 навыки разработки и анализа данных с использованием обоих этих языков: R помо-
834 жет применять самые свежие методы и создавать качественные понятные пользова-
835 телям описания и визуализации, Python пригодится там, где требуется разработка
836 серьёзной промышленной системы, предназначенной для многократного выполне-
837 ния одинаковых задач. В целом же можно повторить основной тезис: данные языки
838 в существенной степени взаимозаменяемы.

839 2.3. Система контроля версий Git

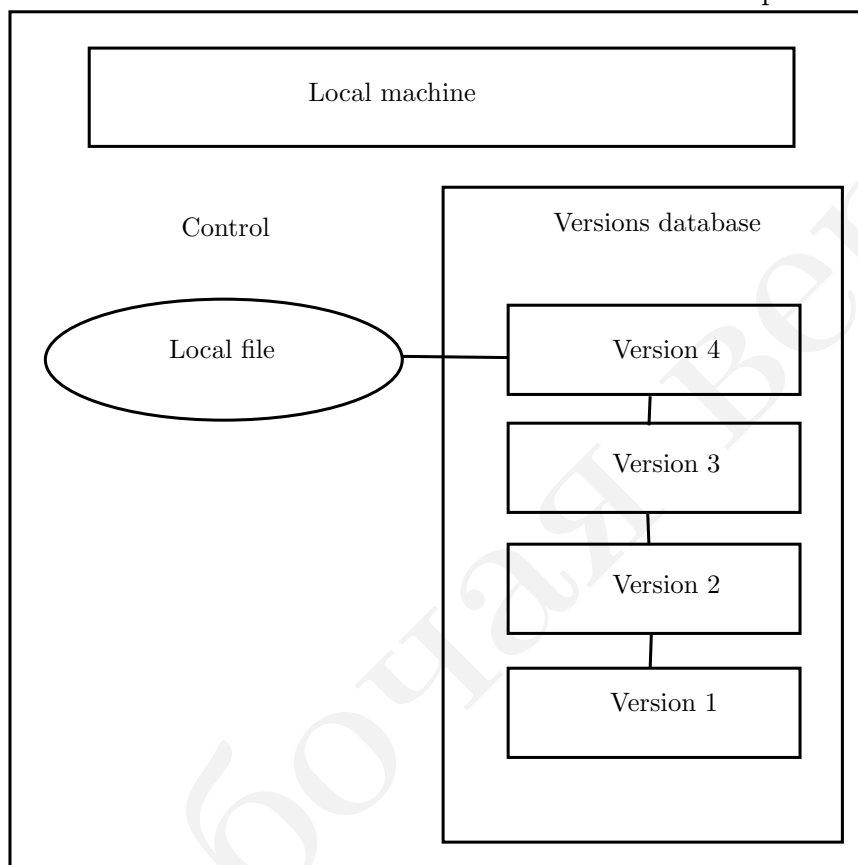
840 2.3.1. Общие сведения

841 Данный раздел не имеет отношения непосредственно к анализу данных, одна-
842 ко содержит сведения, полезные для комфортной работы при его осуществлении.
843 Кроме того, использование систем контроля версий де факто является стандартом
844 при любой серьёзной разработке, особенно в случае совместной работы над одним
845 проектом нескольких аналитиков.

846 Система [Git](#) [21] — это одна из систем контроля версий. Система контроля версий
847 — это система, записывающая изменения в файл или набор файлов в течение време-
848 ни и позволяющая вернуться позже к определённой версии. Как правило подразу-
849 мевается контроль версий файлов, содержащих исходный код программного обеспе-
850 чения, хотя возможен контроль версий практически любых типов файлов [4]. Такие
851 системы позволяют не только хранить версии файлов, но и содержат всю историю
852 их изменения, позволяя отслеживать пошаговое изменение каждого бита файла.
853 Это бывает особенно полезно в тех случаях, когда необходимо иметь возможность
854 «откатить» изменения в случае наличия в них ошибок либо тогда, когда над одним

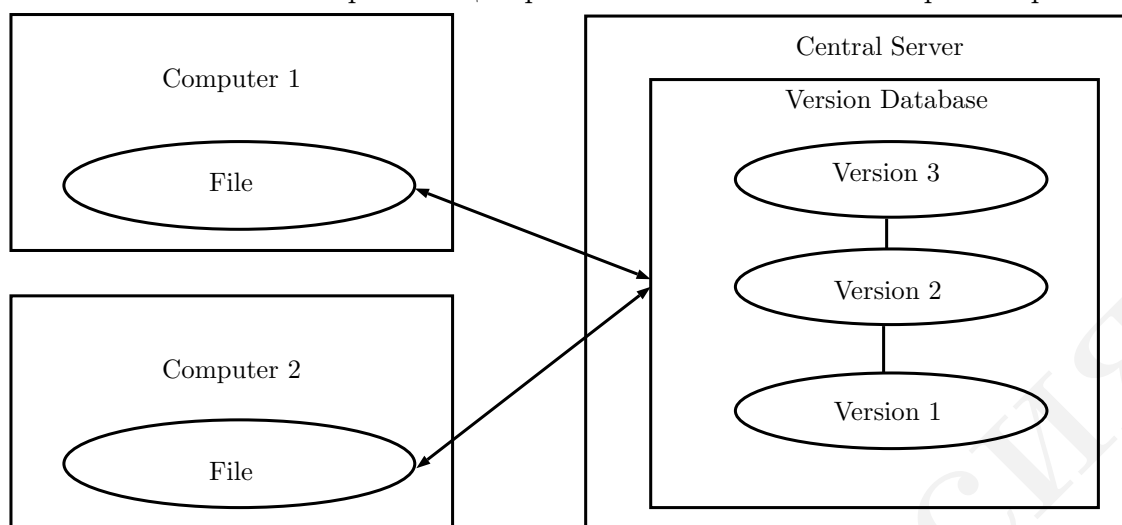
и тем же проектом работает несколько разработчиков либо их команд. Конечно же можно просто создавать полные копии всех файлов проекта. Однако данный способ полезен лишь для создания бэкапов на случай каких-то аварийных ситуаций. В обычной работе он, как минимум, неудобен, а, как максимум, просто не способен обеспечить пошаговое отслеживание изменений файлов и тем более слияние результатов нескольких команд, параллельно работающих над одними и теми же файлами. Для решения данной проблемы были разработаны локальные системы контроля версий, содержащие базу данных всех изменений в файлах, примерная схема организации которых показана на рисунке 2.3.1.

Рис. 2.3.1. Локальная система контроля версий



Современные системы контроля версия бывают централизованными и распределёнными. Первые устроены таким образом, что вся история изменений файлов хранится на центральном сервере, на который пользователи отправляют свои изменения, и с которого они их получают. Общая схема работы централизованной системы контроля версий приведена на рисунке 2.3.2 на следующей странице. Недостатком такой системы является её зависимость от работы центрального сервера. В случае его остановки пользователи не смогут обрабатывать изменения, принимать и отправлять их. Также существует риск полной потери всей истории в случае окончательного отказа сервера.

Рис. 2.3.2. Схема работы централизованной системы контроля версий



Распределённые системы контроля версий лишены данного недостатка, поскольку у каждого пользователя хранится полная история изменений. В связи с этим каждый пользователь может продолжать работать с системой контроля при отсутствии связи с сервером. После восстановления работоспособности последнего, пользователь сможет синхронизировать свою историю изменений с другими разработчиками. Даже в случае полного отказа сервера команда сможет просто перевести хранение на другой и продолжить работу в прежнем режиме. Общая схема работы распределённой системы приведена на рисунке ?? на с. ??.

Особенностью работы системы Git является заложенный в ней принцип работы. В отличие от некоторых других систем контроля версий, принцип которых основан на хранении исходного файла и списка изменений к нему, Git хранит состояние каждого файла после его сохранения, создавая его «снимок». В терминологии Git каждый такой снимок называется commit. При этом создаются ссылки на каждый из файлов. В случае, если при создании нового commit Git обнаруживает, что какие-то файлы не были изменены, система не включает сами файлы в новый commit, а лишь указывает ссылку на последнее актуальное состояние файла из предыдущего commit, обеспечивая таким образом эффективность дискового пространства. При этом каждый commit в целом ссылается на предыдущий, являющийся для него родительским. На рисунке ?? на с. ?? показана общая схема работы системы Git. Линиями со сплошным заполнением показана передача нового состояния файла, возникшего в результате внесения в него изменений, прерывистым — передача ссылки на состояние файла, не подвергавшегося изменениям, из прежнего commit. На момент времени 0 (initial commit) все файлы находились в состоянии 0. Затем в файлы В и С были внесены изменения, тогда как файл А остался в прежнем состоянии. В процессе создания commit № 1 Git сделал снимок состояния файлов В1 и С1, а также создал ссылку на состояние файла А0. Далее изменения были внесены в файл В. В процессе создания commit № 2 Git сохранил состояние файла В2, а также со-

здал ссылки на состояния файлов A0 и C1 в предыдущем commit № 1. Затем были внесены изменения во все три файла, в результате чего на этапе создания commit № 3 Git сделал снимок состояний всех трёх файлов.

Внимательный читатель скорее всего обратил внимание на третий тип линий — пунктир, которому соответствует подпись «hash». Чтобы понять, каким образом в Git реализуется целостность версий, необходимо обратиться к понятию **хеш-функции** [8, 95].

2.3.2. Хеш-функции

Приведём основные определения.

Хеш функция (функция свёртки) — функция, представляющая собой **детерминированный математический алгоритм** [81], осуществляющая преобразование данных произвольной длины в результирующую битовую строку фиксированной длины.

Хеширование — преобразование, осуществляемое хеш-функцией.

Сообщение (ключ, входной массив) — исходные данные.

Хеш (хеш-сумма, хеш-код, сводка сообщения) — результат хеширования.

Согласно **Принципу Дирихле** [88], между хешем и сообщением в общем отсутствует однозначное соответствие. При этом, число возможных значений хеша меньше числа возможных значений сообщения. Ситуация, при которой применение одной и той же хеш-функции к двум различным сообщениям приводит к одинаковому значению хеша, называется «**коллизией хеш функции**» [83]. Т.е. коллизия имеет место тогда, когда $H(x) = H(y)$.

Теоретическая «идеальная» хеш-функция отвечает следующим требованиям:

- а) является детерминированной, то есть её применение к одному и тому же сообщению приводит к одному и тому же значению хеша любое число раз;
- б) значение хеша быстро вычисляется для любого сообщения;
- в) зная значение хеша, невозможно определить значение сообщения;
- г) невозможно найти такие два разных сообщения, применение хеширования к которым приводило бы к одинаковому значению хеша (т.е. идеальная хеш-функция исключает возможность возникновения коллизии);
- е) любое изменение сообщения (вплоть до изменения значения одного бита) изменяет хеш настолько сильно, что новое и старое значения выглядят никак не связанными друг с другом.

Как правило, название хеш-функции содержит значение длины результирующей битовой строки. Например хеш-функция [SHA3-512](#) [76] возвращает строку длиной в 512 бит. Воспользуемся [одним](#) [57] из онлайн-сервисов вычисления хеша и посчитаем его значение для названия данной книги. Как видно на рисунке ?? на с. ??, результатом вычисления хеш-функции является строка длиной в 512 бит, содержащая 128 шестнадцатеричных чисел. При этом, можно наблюдать, что добавление точки в конце предложения полностью меняет значение хеша.

Длина хеша в битах определяет максимальное количество сообщений, для которых может быть вычислен уникальный хеш. Расчёт осуществляется по формуле.

$$2^n \quad (2.3.1)$$

, где n — длина строки в битах.

Так, для функции SHA3-512 число сообщений, имеющих уникальный хеш составляет: $2^{512} \sim 1.340781 \times 10^{154}$. Таким образом, можно говорить о том, что современные хеш-функции способны генерировать уникальный хеш для сообщений любой длины.

Таким образом, Git в процессе создания нового commit сначала вычисляет его хеш-сумму, а затем фиксирует состояние. При этом в каждом commit присутствует ссылка на предыдущий, также имеющий свою хеш-сумму. Таким образом, обеспечивается целостность истории изменений, поскольку значение хеш-суммы каждого последующего commit вычисляется на основе сообщения, содержащего в т. ч. свою хеш-сумму. В этом случае любая модификация содержимого данных, образующих любой commit, неизбежно приведёт к изменению всех последующих хешей, что не останется незамеченным.

2.3.3. Начало работы с Git и основные команды

Для того, чтобы начать работать с Git прежде всего его конечно же следует установить. Как правило, с этим не возникает никаких сложностей. Однако всё же вопросы установки Git кратко рассмотрены в подразделе [2.4.1 Git](#) 47–48.

В данном подразделе преимущественно рассматриваются аспекты работы с ним через командную строку. Данный выбор обусловлен тем обстоятельством, что существует множество графических интерфейсов для работы с Git, которые активно развиваются, меняют дизайн и расширяют функционал. Кроме того, появляются новые продукты. Среди такого разнообразия всегда можно выбрать какой-то наиболее близкий для себя вариант. Таким образом, автор не видит смысла останавливаться на разборе какого-то конкретного графического интерфейса. Более важной задачей является изложение сути и основных принципов работы, понимание которых обеспечит успешную работу с Git безотносительно конкретных программных средств. Кроме того, следует отметить, что практически все современные [IDE](#) [82] имеют свои средства и интерфейс для работы с Git. В дальнейшем в главах, посвящённых непосредственно применению R и Python, будут рассмотрены вопросы использования Git средствами RStudio, Spyder и PyCharm.

В данном подразделе описывается работа с Git через командную строку в операционной системе Kubuntu. Большая часть изложенного применима для любой операционной системы. Для начала работы с Git откроем терминал и выполним три основные настройки, а именно укажем:

- имя пользователя;
- адрес электронной почты;
- текстовый редактор по умолчанию.

Для конфигурации Git существует специальная утилита *git config*, имеющая три уровня глобальности настроек:

- ```
git config --system
```

— системный уровень: затрагивает все репозитории всех пользователей системы;
- ```
git config --global
```

— глобальный уровень: затрагивает все репозитории конкретного пользователя системы;
- ```
git config --local
```

— локальный уровень: затрагивает конкретный репозиторий;

Представим, что необходимо задать общие настройки конкретного пользователя, т.е. использовать уровень *global*, что, может быть актуально, например, при использовании рабочего компьютера. Сделаем следующие настройки:

```
git config --global user.name "First.Second"
git config --global user.email user-adress@host.com
git config --global core.editor "kate"
```

— мы задали имя пользователя, адрес его электронной почты, отображаемые при выполнении *commit*, а также указали текстовый редактор по умолчанию. В данном случае был указан редактор Kate. Естественно можно указать любой другой удобный редактор. В случае использования операционной системы Windows необходимо указывать полный путь до исполняемого файла (имеет расширение *.exe*) текстового редактора, а также а. Например, в случае использования 64-х разрядной Windows и редактора [Notepad++](#) [51] команда может выглядеть так:

```
git config --global core.editor "'C:\Program Files\Notepad\
notepad.exe' -multiInst -notabbar -nosession -noPlugin"
```

1014 — перечень команд для различных операционных систем и текстовых редакторов  
1015 содержится на [соответствующей странице](#) сайта Git [21].

1016 Для начала создадим тестовый каталог, с которым и будем работать в дальней-  
1017 шем при обучении работе с Git. Зайдём в папку, в которой хотим создать каталог  
1018 и запустим терминал в ней. После чего введём команду:

```
1019
1020 mkdir git-lesson
1021
```

1022 — мы только что создали новый каталог средствами командной строки.

1023 Затем введём команду:

```
1024
1025 cd git-lesson
1026
```

1027 — переходим в только что созданный каталог.

1028 Для просмотра содержимого каталога используем следующую команду:

```
1029
1030 ls -la
1031
```

1032 — собственно самой командой является `ls`, а «`-la`» представляет собой её аргу-  
1033 менты: «`-l`» — отвечает за отображение файлов и подкаталогов списком, а «`-a`» —  
1034 за отображение скрытых файлов и подкаталогов.

1035 Для создания репозитория введём команду:

```
1036
1037 git init
1038
```

1039 — Git ассоциирует текущую папку с новым репозиторием.

1040 В случае, если всё прошло хорошо, терминал возвратит следующее сообщение:

```
1041
1042 Initialized empty Git repository in /home/.../git-lesson/.
1043 git/
1044
```

1045 Теперь ещё раз введём:

```
1046
1047 ls -la
1048
```

1049 — следует обратить внимание на то, что появилась папка `.git`, в которой и будет  
1050 храниться вся история версий проекта, содержащегося в папке `git-lesson`.

1051 Создадим первый файл внутри папки:

```
1052
1053 touch file1.py
1054
```

1055 — расширение указывает на то, что это файл языка Python.

1056 Система Git уже должна была отследить наличие изменения состояния проекта,  
1057 произошедшее вследствие создания нового файла. Для проверки изменений состо-  
1058 яния используем команду:

```
1059
1060 git log
1061
```

1062 — и получим сообщение следующего содержания:

```
1063
1064 fatal: your current branch 'master' does not have any
1065 commits yet
1066
```

1067 — дело в том, что в истории изменений по-прежнему нет никаких записей.

1068 Для получения дополнительных сведений используем команду:

```
1069 git status
```

1072 — терминал возвратит следующее сообщение:

```
1073 On branch master
1074
1075
1076 No commits yet
1077
1078 Untracked files:
1079 (use "git add <file>..." to include in what will be
1080 committed)
1081 file1.py
1082
1083 nothing added to commit but untracked files present (use "
1084 git add" to track)
```

1086 — как видно, Git сообщает о том, что файл `file1.py` не отслеживается, кроме того, как следует из последней части сообщения терминала, в настоящее время вообще не фиксируются никакие изменения, поскольку ничего не было добавлено в лист отслеживания. При этом сам Git предлагает использовать команду `git add` для добавления файлов в него. Прежде чем сделать это, необходимо разобраться в том, в каких состояниях, с точки зрения Git, могут в принципе находиться файлы.

1092 Все файлы, находящиеся в рабочем каталоге, могут иметь один из следующих статусов:

- 1094 • `tracked` — отслеживаемые, т. е. находящиеся под версионным контролем;
- 1095 • `untracked` — не отслеживаемые, т. е. не находящиеся под версионным контролем.

1097 Ко второй категории, как правило, относятся временные файлы, например логи, хранение которых в репозитории нецелесообразно. Файлы первой категории могут находиться в одной из следующих состояний:

- 1100 • `initial` — начальное состояние файла, в котором он находился в момент включения его в лист отслеживания, т. е. сообщения ему статуса `tracked`.
- 1102 • `modified` — состояние файла после внесения в него изменений и его сохранения;
- 1103 • `staged` — промежуточное состояние файла, в котором он находится после передачи его состояния Git, но до формирования последним его снимка.
- 1105 • `committed` — состояние файла, зафиксированное Git, и представляющее его версию, к которой впоследствии будет возможно вернуться.

1107 Соответственно после внесения новых изменений файл, находящийся в состоянии  
 1108 committed, переходит в состояние modified, после чего возможен новый цикл преоб-  
 1109 разований его статуса. Схема изменений состояния файлов приведена на рисунке ??  
 1110 на с. ??.

1111 Для перевода файла из состояния modified в состояние staged следует использо-  
 1112 вать команду

```
1113 git add <file.name1> <file.name2>
```

1116 — данная процедура также называется добавлением файла в индекс. Индекс — об-  
 1117 ласть памяти, в которой находятся файлы, подготовленные для включения в commit.

1118 Далее для выполнения процедуры commit даётся команда

```
1119 git commit -m "message"
```

1122 — аргумент -m и следующее за ним сообщение служат для задания краткого опи-  
 1123 сания того, какие изменения были внесены. Рекомендуется давать содержательные  
 1124 комментарии, позволяющие понять смысл изменений.

1125 Как видно, не обязательно совершать процедуру commit сразу в отношении всех  
 1126 файлов, находящихся в состоянии modified. Существует возможность группировать  
 1127 их и, посредством перевода конкретных файлов в состояние staged, формировать  
 1128 группы файлов, чьё состояние подлежит фиксации.

1129 Добавим файл file.py в индекс.

```
1130 git add file1.py
```

1133 Далее снова проверим статус:

```
1134 git status
```

1137 — на этот раз терминал возвратит новое сообщение:

```
1138 On branch master
1139
1140
1141 No commits yet
1142
1143 Changes to be committed:
1144 (use "git rm --cached <file>..." to unstage)
1145 new file: file1.py
```

1147 Как можно видеть, теперь Git «видит» файл file1.py и готов сделать «снимок» но-  
 1148 вого состояния репозитория. Для выполнения процедуры commit введём команду:

```
1149 git commit -m "First commit"
```

1152 — мы только что сделали первый commit, т.е. зафиксировали состояние репозито-  
 1153 рия. Терминал возвратит следующее сообщение:

```
1154 [master (root-commit) 1306b16] First commit
1155 1 file changed, 0 insertions(+), 0 deletions(-)
```

```
1157 create mode 100644 file1.py
1158
```

1159 Теперь повторим ранее уже использованную команду:

```
1160 git log
1161
1162
```

1163 — терминал в отличие от первого раза, когда мы наблюдали сообщение о невоз-  
1164 можности вывода сведений о событиях в репозитории, на этот раз возвращает  
1165 осмысленное сообщение:

```
1166 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->
1167 master) Author: Kirill Murashev <kirill.murashev@gmail.
1168 com>
1169 Date: Tue Aug 31 19:03:49 2021 +0200
1170 First commit
1171
1172
```

1173 — можно увидеть хеш-сумму данного commit, его автора, а также время созда-  
1174 ния commit и сопроводительное сообщение к нему. Для получения более детальных  
1175 сведений можно использовать команду `git show`, сообщив ей в качестве аргумен-  
1176 та хеш-сумму интересующего commit. Сделаем это, скопировав и вставив значение  
1177 хеш-суммы:<sup>4</sup>

```
1178 git show 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1179
1180
```

1181 — в качестве аргумента команды в данном случае была использована хеш-сумма.  
1182 Терминал возвратит сообщение с данными об интересующем commit:

```
1183 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->
1184 master)
1185 Author: Kirill Murashev <kirill.murashev@gmail.com>
1186 Date: Tue Aug 31 19:03:49 2021 +0300
1187
1188 First commit
1189
1190 diff --git a/file1.py b/file1.py
1191 new file mode 100644
1192 index 0000000..e69de29
1193
1194
```

1195 В дополнение к уже имеющимся данным приводятся сведения о том, какие имен-  
1196 ные изменения имели место. В данном случае видно, что имело место добавление  
1197 в репозиторий нового файла.

1198 Примерно такие же сведения можно получить в случае использования команды  
1199 `git log` с аргументом `-p`.

```
1200 $ git log -p
1201
1202
```

<sup>4</sup>Для копирования и вставки в окне терминала следует использовать сочетания клавиш `ctrl+shift+c`, `ctrl+shift+v` соответственно.



```

1203 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->
1204 master) Author: Kirill Murashev <kirill.murashev@gmail.
1205 com> Date: Tue Aug 31 19:03:49 2021 +0300
1206
1207 First commit
1208
1209 diff --git a/file1.py b/file1.py
1210 new file mode 100644
1211 index 0000000..e69de29
1212

```

1213 — в данном случае сообщения вообще идентичны.

1214 Рассмотрим ещё одну полезную команду `git restore`. Данная команда возвра-  
 1215 щает состояние файла к тому состоянию, которое было зафиксировано при создании  
 1216 последнего commit. Рассмотрим пример. Откроем файл `file1.py` в редакторе Kate<sup>5</sup>  
 1217 непосредственно из терминала:

```

1218 kate file.py
1219
1220

```

1221 — далее напишем в нём любой текст и сохраним файл. После чего проверим его ста-  
 1222 тус с помощью уже известной команды `git status`:

```

1223 $git status
1224
1225 On branch master
1226 Changes not staged for commit:
1227 (use "git add <file>..." to update what will be committed
1228)
1229 (use "git restore <file>..." to discard changes in working
1230 directory)
1231 modified: file1.py
1232
1233 no changes added to commit (use "git add" and/or "git commit
1234 -a")
1235
1236

```

1237 — как видим, Git обнаружил изменение файла. Теперь введём команду:

```

1238 git restore file.py
1239
1240

```

1241 — файл, возвращён в состояние, в котором он находился на момент создания по-  
 1242 следнего commit, т. е. снова является пустым, в чём легко убедиться, открыв его.

1243 Следующей рассматриваемой командой будет `git diff`. Данная команда позво-  
 1244 лят понять, какие именно изменения были внесены в файл. Вновь откроем файл  
 1245 `file1.py` в текстовом редакторе. Введём в него текст, например «Liberte, egalite,  
 1246 fraternite». После чего сохраним файл. Выполним команду `git diff` и посмотрим  
 1247 на результат.

<sup>5</sup>Естественно редактор может быть любой



```

1248 $git diff
1249
1250
1251 diff --git a/file1.py b/file1.py
1252 index e69de29..72d6a2a 100644
1253 --- a/file1.py
1254 +++ b/file1.py
1255 @@ -0,0 +1 @@
1256 +Liberte, egalite, fraternite
1257

```

1258 — в нижней части сообщения терминала после символа «+» мы видим добавленный  
 1259 в файл текст. Git всегда отображает добавленный текст после знака «+», а удалённый  
 1260 после знака «-». Проверим статус файла:

```

1261 $ git status
1262
1263
1264 On branch master
1265 Changes not staged for commit:
1266 (use "git add <file>..." to update what will be committed)
1267 (use "git restore <file>..." to discard changes in working
1268 directory)
1269 modified: file1.py
1270
1271 no changes added to commit (use "git add" and/or "git commit
1272 -a")
1273

```

1274 — Git зафиксировал изменения файла. Теперь добавим файл в индекс, т. е. изменим  
 1275 его состояние на staged:

```

1276 git add file1.py
1277
1278

```

1279 — далее ещё раз проверим статус файла:

```

1280 $ git status
1281
1282
1283 On branch master
1284 Changes to be committed:
1285 (use "git restore --staged <file>..." to unstage)
1286 modified: file1.py
1287

```

1288 — Git перевёл файл в состояние staged. Для того, чтобы ещё раз посмотреть изме-  
 1289 нения в файле, находящемся в состоянии staged можно использовать ту же команду  
 1290 git diff, при условии сообщения ей аргумента --staged, без которого она не смо-  
 1291 жет отобразить изменения, поскольку они уже были включены в индекс.

```

1292 $git diff --staged
1293
1294
1295 diff --git a/file1.py b/file1.py

```

```

1296 index e69de29..d77d790 100644
1297 --- a/file1.py
1298 +++ b/file1.py
1299 @@ -0,0 +1 @@
1300 +Liberte, egalite, fraternite
1301

```

1302 Выполним commit:

```

1303
1304 git commit -m "Second commit"
1305

```

1306 — терминал возвратит сообщение:

```

1307 [master 700a993] Second commit
1308 1 file changed, 1 insertion(+)
1309
1310

```

1311 — посмотрим на историю изменений:

```

1312 $ git log
1313
1314
1315 commit 700a993db7c5f682c33a087cb882728adc485198 (HEAD ->
1316 master)
1317 Author: Kirill Murashev <kirill.murashev@gmail.com>
1318 Date: Tue Aug 31 20:51:06 2021 +0200
1319
1320 Second commit
1321
1322 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1323 Author: Kirill Murashev <kirill.murashev@gmail.com>
1324 Date: Tue Aug 31 19:03:49 2021 +0200
1325
1326 First commit
1327

```

1328 — можно наблюдать сведения о двух выполненных commit.

1329 В случае использования той же команды с аргументом -p можно увидеть всю  
 1330 историю конкретных изменений.

```

1331 $ git log -p
1332
1333 commit 700a993db7c5f682c33a087cb882728adc485198 (HEAD ->
1334 master)
1335 Author: Kirill Murashev <kirill.murashev@gmail.com>
1336 Date: Tue Aug 31 20:51:06 2021 +0300
1337
1338 Second commit
1339
1340 diff --git a/file1.py b/file1.py
1341 index e69de29..d77d790 100644
1342 --- a/file1.py

```

```

1343 +++ b/file1.py
1344 @@ -0,0 +1 @@
1345 +Liberte, egalite, fraternite
1346
1347 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1348 Author: Kirill Murashev <kirill.murashev@gmail.com>
1349 Date: Tue Aug 31 19:03:49 2021 +0300
1350 First commit
1351 diff --git a/file1.py b/file1.py
1352 new file mode 100644
1353 index 0000000..e69de29
1354

```

1355 Существует упрощённый способ передачи Git сведений для совершения commit.  
 1356 Вместо последовательного ввода команд `git add` с указанием перечня файлов и `git`  
 1357 `commit` можно использовать единую команду `git commit` с аргументами `-am`. Вто-  
 1358 рой аргумент, как уже было сказано ранее, необходим для формирования сообще-  
 1359 ния, сопровождающего commit. Первый же заменяет собой предварительное ис-  
 1360 пользование команды `git add`, указывая Git на необходимость включения в индекс  
 1361 всех отслеживаемых файлов, т. е. имеющих статус `tracked`. Внесём любые изменения  
 1362 в файл `file1.py`. Проверим наличие изменений:

```

1363 $ git status
1364
1365 On branch master
1366 Changes not staged for commit:
1367 (use "git add <file>..." to update what will be committed)
1368 use "git restore <file>..." to discard changes in working
1369 directory)
1370 modified: file1.py
1371
1372 no changes added to commit (use "git add" and/or "git commit
1373 -a")
1374
1375

```

1376 — после чего выполним добавление в индекс и commit одной командой.

```

1377 $ git commit -am "Third commit"
1378 [master fbff919] Third commit
1379 1 file changed, 1 insertion(+)
1380
1381

```

1382 — проверим историю:

```

1383 $ git log -p
1384
1385
1386 commit fbff919fab14ab6d41c993d3b86253c41037e075 (HEAD ->
1387 master)
1388 Author: Kirill Murashev <kirill.murashev@gmail.com>
1389 Date: Tue Aug 31 21:25:45 2021 +0300

```

```

1390
1391 Third commit
1392
1393 diff --git a/file1.py b/file1.py
1394 index d77d790..bf6409f 100644
1395 --- a/file1.py
1396 +++ b/file1.py @@ -1 +1,2 @@
1397 Liberte, egalite, fraternite
1398 +Жизнь, свобода, собственность
1399
1400 commit 700a993db7c5f682c33a087cb882728adc485198
1401 Author: Kirill Murashev <kirill.murashev@gmail.com>
1402 Date: Tue Aug 31 20:51:06 2021 +0300
1403
1404 Second commit
1405
1406 diff --git a/file1.py b/file1.py
1407 index e69de29..d77d790 100644
1408 --- a/file1.py
1409 +++ b/file1.py
1410 @@ -0,0 +1 @@
1411 +Liberte, egalite, fraternite
1412
1413 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1414 Author: Kirill Murashev <kirill.murashev@gmail.com>
1415 Date: Tue Aug 31 19:03:49 2021 +0300
1416
1417 First commit
1418
1419 diff --git a/file1.py b/file1.py
1420 new file mode 100644
1421 index 0000000..e69de29
1422

```

1423 — можно наблюдать уже три commit.

1424 Следующей полезной командой является `git mv`. Данная команда позволяет, в част-  
 1425 ности, переименовывать либо перемещать файлы. При этом её выполнение автома-  
 1426 тически переводит файл в состояние staged, минуя состояние modified. Выполним  
 1427 переименование:

```

1428
1429 $ git mv file1.py file-1.py
1430

```

1431 — затем проверим состояние:

```

1432
1433 $ git status
1434
1435 On branch master

```

```

1436 Changes to be committed:
1437 (use "git restore --staged <file>..." to unstage)
1438 renamed: file1.py -> file-1.py
1439

```

1440 — как можно увидеть, файл с новым именем готов к commit. Выполним commit.

```

1441 $ git commit -m "Fourth commit"
1442
1443 [master 284073c] Fourth commit
1444 1 file changed, 0 insertions(+), 0 deletions(-)
1445 rename file1.py => file-1.py (100%)
1446
1447

```

1448 — изменения файла зафиксированы.

1449 Следующей заслуживающей внимания командой является `git rm`. Данная ко-  
 1450 манда удаляет файл.

```

1451 git rm file-1.py
1452
1453

```

1454 — проверим выполнение операции:

```

1455 $ git status
1456
1457 On branch master
1458 Changes to be committed:
1459 (use "git restore --staged <file>..." to unstage)
1460 deleted: file-1.py
1461
1462

```

1463 — как видно из сообщения Git в терминале, существует возможность восстановить  
 1464 удалённый файл в том состоянии, которое было зафиксировано при выполнении  
 1465 последнего commit. Выполним команду для восстановления файла:

```

1466 git restore --staged file-1.py
1467
1468

```

1469 — затем проверим его состояние:

```

1470 $ git status
1471
1472 On branch master
1473 Changes not staged for commit:
1474 (use "git add/rm <file>..." to update what will be
1475 committed)
1476 (use "git restore <file>..." to discard changes in working
1477 directory)
1478 deleted: file-1.py
1479
1480
1481 no changes added to commit (use "git add" and/or "git commit
1482 -a")
1483

```

1484 — как следует из сообщения Git, файл file-1.py больше не находится в индексе,  
 1485 для его возвращения туда необходимо выполнить команду `git restore` без указа-  
 1486 ния каких-либо аргументов.

```
1487
1488 git restore file-1.py
1489
```

1490 — ещё раз проверим состояние:

```
1491 $ git status
1492
1493
1494 On branch master nothing to commit, working tree clean
1495
```

1496 — файл снова включён в индекс, его состояние соответствует состоянию, зафиксиро-  
 1497 ванному при выполнении последнего commit. Сам файл при этом вновь присут-  
 1498 ствует в каталоге.

1499 Команда `git rm` также может быть использована для передачи файлу статуса  
 1500 untracked без его удаления из каталога. Для этого ей необходимо сообщить аргумент  
 1501 `--cached`.

```
1502 $ git rm --cached file-1.py
1503
1504
1505 rm 'file-1.py'
1506
```

1507 — файл был исключён из индекса, а также из списка отслеживания, но при этом  
 1508 остался в каталоге, в чём можно легко убедиться:

```
1509 $ git status
1510 On branch master
1511 Changes to be committed:
1512 (use "git restore --staged <file>..." to unstage)
1513 deleted: file-1.py
1514
1515 Untracked files:
1516 (use "git add <file>..." to include in what will be
1517 committed)
1518 file-1.py
1519
1520
```

1521 — есть изменения, доступные для commit, а также в каталоге присутствует неот-  
 1522 слеживаемый файл (статус untracked).

```
1523 $ ls -la
1524 total 0\
1525
1526 drwx----- 1 kaarlahti root 0 jaan 1 1970 .
1527 drwx----- 1 kaarlahti root 0 jaan 1 1970 ..
1528 -rwx----- 1 kaarlahti root 84 sept 1 19:08 file-1.py
1529 drwx----- 1 kaarlahti root 0 jaan 1 1970 .git
1530
```

1531 — файл присутствует в каталоге.

1532 Выполним commit:

```

1533 $ git commit -m "Fifth commit"
1534 [master 7abee55] Fifth commit
1535 1 file changed, 2 deletions(-)
1536 delete mode 100644 file-1.py
1537
1538

```

1539 EndEnd

#### 1540 2.3.4. Исключение файлов из списка отслеживания

#### 1541 2.3.5. Ветки проекта, указатели

#### 1542 2.3.6. Работа с ГитХаб

#### 1543 2.3.7. Rebase

#### 1544 2.3.8. Работа с Git в IDE

1545 End

### 1546 2.4. Установка и настройка

#### 1547 2.4.1. Git

##### 1548 2.4.1.1. Установка на операционных системах, основанных на Debian: 1549 Debian, Ubuntu, Mint и т. п.

1550 В операционных системах, основанных на ядре Linux [52], относящихся к ветке  
1551 Debian [71], Git зачастую бывает уже установлен вместе с системой. Чтобы прове-  
1552 рить наличие Git в командную строку терминала следует ввести:

```

1553 git
1554
1555

```

1556 В случае наличия Git в системе, терминал возвратит длинное сообщение, начина-  
1557 ющееся примерно следующим образом:

```

1558 usage: git [--version] [--help] [-C <path>] [-c <name>=<
1559 value>]
1560 [--exec-path[=<path>]] [--html-path] [--man-path] [--info-
1561 path] [-p | --paginate | -P | --no-pager] [--
1562 no-replace-objects] [--bare] [--git-dir=<path
1563 >] [--work-tree=<path>] [--namespace=<name>]
1564
1565

```

1566 В случае его отсутствия:

```

1567 Command 'git' not found, did you mean:
1568
1569

```

1570 Во втором случае следует использовать следующие команды:



```
1571 sudo apt update -y
1572
1573 sudo apt install git -y
1574
```

1575 Процесс проходит автоматически и не требует внимания со стороны пользователя.

#### 1576 2.4.1.2. Установка на операционной системе Windows

1577 Установка Git на Windows осуществляется обычным для данной операционной  
1578 системы образом. Необходимо загрузить установочный файл с [соответствующей](#)  
1579 [страницы](#) [19] и запустить процесс установки, желательно приняв при этом все на-  
1580 стройки по умолчанию.

#### 1581 2.4.1.3. Установка на macOS

1582 Существует несколько способов установки Git на macOS. Их перечень приведён  
1583 на [соответствующей странице](#) [20] сайта Git. Следует отметить, что в случае нали-  
1584 чия в системе Xcode [96] Git также уже присутствует, и его установка не требуется.  
1585 В данном материала приводится один из возможных способов. Для начала необхо-  
1586 димо установить менеджер пакетов Homebrew [25]. Для этого в командной строке  
1587 терминала необходимо ввести следующую команду:

```
1588 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com
1589 /Homebrew/install/HEAD/install.sh)"
1590
1591
```

1592 После этого можно перейти к установке самого Git. Для этого в командной строке  
1593 терминала необходимо ввести следующую команду:

```
1594 brew install git
1595
1596
```

1597 Как и в случае, описанном выше в секции 2.4.1.1 на предыдущей странице–48, про-  
1598 цесс проходит автоматически и не требует внимания со стороны пользователя.

1599 R  
1600 RStudio  
1601 Python  
1602 End  
1603 The End

,III,Ч,II,X,

Рабочая версия