**РОСЖЕЛДОР**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

|  |  |  |  |
| --- | --- | --- | --- |
| **К защите:** |  |  | |
| **Заведующий кафедрой** | **Информационные** | |
| **технологии транспорта** | | |
|  | д–р техн. наук, проф. | |
|  |  | В. И. Хабаров | |
| *подпись* |  | *инициалы, фамилия* | |
|  |  |  | |
| *дата* |  |  | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тема:** | Использование машинного обучения в задачах программной инженерии | | | | | |
|  |  | | | | | |
|  | |  | МД.МИСТ.05.2022 |  |  |
|  | |  | *шифр документа* |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Выполнил** |  |  |  | **Руководитель** |
|  |  | Н.Н. Насридинова |  |  |  | канд. техн. наук, доц.  Е.Б. Тарасов |
| *Подпись* |  | *инициалы, фамилия* |  | *Подпись* |  | *инициалы, фамилия* |
|  |  |  |  |  |  |  |
| *Дата* |  |  |  | *Дата* |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Консультанты по разделам** |  |  |  |  |
| 1 |  |  |  |  |
|  |  |  |  |  |
| 2 |  |  |  |  |
|  |  |  |  |  |
| 3 |  |  |  |  |
|  |  |  |  |  |
| Нормоконтролер работы |  |  |  | ст. преп.  Т.А. Распопина |
|  |  | *подпись* |  |  |
|  |  |  |  |  |
|  |  | *Дата* |  |  |

**2022 г.**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес–информатики

Кафедра: Информационные технологии транспорта

Направление: 09.04.02 «Информационные системы и технологии»

Программа: Интеллектуальные транспортные системы

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д–р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| Студенту | Насридиновой Наргис Назрихудоевне | | |
|  |  | | |
| 1. Тема «Использование машинного обучения в задачах программной инженерии» утверждена приказом № 180/с от «16» мая 2022 г. | | | |
| 2. Задание выдано «28» марта 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «25» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно–пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 2 | 30.03.2022 |
| Аналитическое исследование | | 16 | 05.04.2022 |
| Проектирование информационной системы | | 19 | 15.04.2022 |
| Практическая реализация | | 25 | 28.05.2022 |
| Заключение | | 1 | 03.06.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  листов  формата А1 | График  (сроки)  выполнения |
| Презентация PowerPoint | 20 | 30.05.2022 |

7. Консультанты по разделам

|  |  |  |
| --- | --- | --- |
| Наименование  Раздела | Фамилия, И. О.  Консультанта | Подпись консультанта,  дата выдачи задания |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | Е.Б. Тарасов |
|  | *(подпись, фамилия, И.О.)* |  |
| Задание к использованию принял |  | Н.Н. Насридинова |
|  | *(подпись студента)* |  |

УДК 004.8

**АННОТАЦИЯ**

В работе 67 страниц, 14 рисунков, 1 таблица, 20 источников.

Ключевые слова: *машинное обучение, программная инженерия, программное обеспечение, искусственный интеллект.*

Предметная область – тематическое исследование программной инженерии и использование рабочих процессов машинного обучения.

**ABSTRACT**

The work contains 67 pages, 14 drawings, 1 table, 20 sources.

Keywords: *machine learning, software engineering, software, artificial intelligence.*

Subject area – Software engineering case study and use of machine learning workflows.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

ПрО – предметная область.

МО – машинное обучение.

ИИ – искусственный интеллект.

ПО – программное обеспечение.

ПИ – программная инженерия.

ЭВМ – электронно–вычислительная машина.

ПС – программная система.

ТЗ – техническое задание.

ТП – технология программирования.

ПП – программный продукт.

ИТ – информационные технологии.

ЖЦ – жизненный цикл.

IDE – интегрированная среда разработки.

NLP – нейролингвистическое программирование.

**СОДЕРЖАНИЕ**

[Введение 7](#_Toc107225526)

[1 Машинное обучение 9](#_Toc107225527)

[1.1 История возникновения термина машинное обучение 9](#_Toc107225528)

[1.2 Исследования применения машинного обучения 9](#_Toc107225529)

[1.2.1 Аналогия человеческого мышления и сбор данных машины для создания программных алгоритмов 9](#_Toc107225530)

[1.2.2 Принятие решений на основе данных 14](#_Toc107225531)

[1.3 Рабочий процесс: от данных до внедрения 19](#_Toc107225532)

[1.4 Усовершенствованные способы повышения эффективности 23](#_Toc107225533)

[1.4.1 Предварительная обработка данных и проектирование признаков 23](#_Toc107225534)

[1.4.2 Непрерывное совершенствование моделей 25](#_Toc107225535)

[1.4.3 Масштабирование моделей 26](#_Toc107225536)

[2 Основы программной инженерии 27](#_Toc107225537)

[2.1 Определение базовых понятий программной инженерии 27](#_Toc107225538)

[2.2 Основные объекты разработки программной инженерии 32](#_Toc107225539)

[3 Машинное обучение в программной инженерии 39](#_Toc107225540)

[3.1 Процессы разработки программного обеспечения 42](#_Toc107225541)

[3.2 Рабочий процесс МО 43](#_Toc107225542)

[3.3 Программная инженерия для машинного обучения 46](#_Toc107225543)

[3.3.1 Зрелость процесса 47](#_Toc107225544)

[3.3.2 Исследование 48](#_Toc107225545)

[3.4 Применение искусственного интеллекта 49](#_Toc107225546)

[4 Практики машинного обучения в программной инженерии 51](#_Toc107225547)

[4.1 Доступность, сбор, очистка и управление данными 52](#_Toc107225548)

[4.2 Образование и профессиональная подготовка 53](#_Toc107225549)

[4.3 Эволюция, оценка и внедрение модели 54](#_Toc107225550)

[4.3.1 Разнообразное восприятие 55](#_Toc107225551)

[4.3.2 На пути к модели зрелости процесса MО 57](#_Toc107225552)

[4.3.3 Ограничения 63](#_Toc107225553)

[Заключение 64](#_Toc107225554)

[Список использованных источников 66](#_Toc107225555)

# ВВЕДЕНИЕ

Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов. Машинное обучение как правило, совмещает в себе равные доли математических принципов и полученных эмпирическим путем приемов. Слишком сильная концентрация только на одном аспекте в ущерб другому – проигрышная стратегия. Необходим баланс. Долгое время лучшим способом постижения машинного обучения было получение ученой степени в одной из областей, которые развивали статистические методы и техники оптимизации. Основной упор эти программы делали на ключевые алгоритмы, в том числе на их теоретические свойства и ограничения, а также на характерные особенности относящихся к данной сфере задач. Также, наряду с этим необходимые знания накапливались другим образом – в процессе общения на конференциях, обмена информацией и сценариями обработки данных между специалистами–исследователями. Накопленные знания, позволили установить, оптимальные алгоритмы для каждой ситуации, как поэтапно обрабатывать данные связать между собой различные этапы рабочего процесса [1].

Программная инженерия впервые появилась во второй половине 20 века и заняла одной из основных связующих звеньев информатики, информационных систем и технологий. ПИ охватывает различные отрасли разработки программного обеспечения, в виде продуктов прикладных и информационных систем. Основу положения ПИ в современной науке представляет ядро знаний SWEBOK (Software Engineering body of Knowledge, www.swebok.com, 2001 г.), утвержденным международным комитетом IEEE и ACM. В SWEBOK также сформулировано основное понятие ПИ. ПИ – это система методов, средств и дисциплин планирования, разработки, эксплуатации и сопровождения ПО, подготовленного к внедрению. К основным концепциям ПИ отнесены – продуктивность, индустрия и качество. Это ядро состоит из 10 разделов знаний (area knowledge). Первые пять разделов ядра включают определения требований, проектирования, конструирования, тестирования и сопровождения продуктов ПО. Последующие пять разделов являются организационными. Они включают – управление проектом, конфигурацией, качеством, методы и средства инженерии ПО. Ядру знаний соответствуют процессы жизненного цикла стандарта ISO/IEC 12207, которые ориентированы на реализацию ПО, в соответствии с требованиями определенных разделов SWEBOK [2].

Последние достижения в области МО вызвали широкий интерес в секторе ИТ к интеграции возможностей ИИ в ПИ. Постоянные последовательные сдвиги в области ИТ влекут за собой новые цели в ПИ, которые побуждают многие организации занимающийся разработкой ПО, совершенствовать свои методы разработки для решения новых аспектов этой области. Последняя тенденция в данной индустрии связана с интеграцией возможностей ИИ на основе достижений в области МО. ИИ в широком смысле включает в себя технологии рассуждения, решения проблем, планирования и обучения. В контексте последних тенденций в области МО командой компании Microsoft с применением МО были созданы следующие приложения: Bing Search или виртуальный помощник Cortana, платформа Microsoft Translator для перевода текста, голоса и видео в режиме реального времени, Cognitive Services для понимания зрения, речи и языка для создания интерактивных, разговорных агентов и платформа Azure AI, позволяющая клиентам создавать собственные приложения МО [3].

Объект исследования – использование машинного обучения в программной инженерии.

Предмет исследования – тематическое исследование программной инженерии и использование рабочих процессов машинного обучения.

Цель работы – интеграция возможностей искусственного интеллекта с применением методов машинного обучения в программной инженерии.

# Машинное обучение

## История возникновения термина машинное обучение

В 1959 г. специалист по вычислительной технике из американской компании IBM Артур Самуэль написал компьютерную программу для игры в шашки. Каждому положению на доске присваивался некий вес, базирующийся на вероятности выигрыша. Изначально вероятность определялась по формуле, в которой учитывались такие факторы, как количество шашек на каждой стороне и количество дамок. Подход работал, но Самуэль придумал, ка­ким образом можно повысить его эффективность. Сыграв с программой тысячу партий, он использовал их результаты для уточнения позици­онных весов. Десять лет спустя программа достигла уровня хорошо подготовленного непрофессионального игрока.

Самуэль написал компьютерную программу, которая по мере накопления собственных результатов могла повышать эффективность своего применения. Программа, как бы постоянно училась, так зародилось машинное обучение (ML – machine learning) [4].

## Исследования применения машинного обучения

### **Аналогия человеческого мышления и сбор данных машины для создания программных алгоритмов**

Из исторического опыта выявлено два основополагающих различия обучения, это механическое заучивание и интеллектуальное осмысление информации человеком.

Группой иностранных ученых под руководством Хенрик Бринк была проведена работа по изучению способов восприятия человеком информации и его использования в создании программных тестов:

* на примере ребенка, который играет с друзьями, и наблюдает реакцию других членов группы на свои действия. Этот опыт ребенка влияет на его будущее поведение в социуме. Но он не вспоминает и не проигрывает заново свое прошлое, а опирается на определенные, легко опознаваемые характеристики прошлых взаимодействий: детская площадка, класс, мама, папа, сестры и братья, друзья, незнакомцы, взрослые, дети, в помещении или на улице. Оценка новой ситуации базируется на признаках, с которыми ему доводилось сталкиваться раньше. Обучение при этом является не просто сбором информации. Формируется то, что можно назвать аналитической оценкой;
* на другом примере, представлено, как по картинкам ребенок обучается различать собаку от кошки. Показанная картинка кладется в одну из двух стопок, в зависимости от правильности полученного ответа. Чем дольше продолжается процесс, тем выше эффективность распознавания. Человеческое сознание обладает встроенными механизмами классификации. Ему требуются только образцы. Научившись работать с картинками, ребенок сможет опознать практически любое изображение кошки или собаки. Эта способность обобщать, применяя полученные в процессе тренировок знания к новым, ранее не встречавшимся образцам, является ключевой характеристикой человеческого обучения, также он может быть применен при составлении алгоритмов для машинного обучения;
* проведен конкурс на алгоритм, максимально точно отличающий собак от кошек. Для тренировки участникам предоставили 25000 изображений с метками, указывающими, кто именно изображен на картинке. После обучения каждый алгоритм должен был классифицировать 12500 не имеющих меток тестовых изображений. Возвращаясь к особенностям работы человеческого мышления возникают признаки, по которым можно отличить собаку от кошки. У кошек треугольные и стоячие уши, а у собак они висят, но бывают и исключения.

Для обучения и обобщения люди используют различные данные из примеров, включая формы, цвета, текстуры, пропорции и другие характеристики. Машинное обучение также применяет множество стратегий в различных комбинациях в зависимости от поставленной задачи. Эти стратегии базируются на основе наборов алгоритмов, разработанных в течение последних десятилетий как учеными, так и практиками в самых разных дисциплинах – от статистики, компьютерной науки, робототехники и прикладной математики до поиска в Интернете, развлекательной сферы, цифровой рекламы и переводов с одного языка на другой. Алгоритмы крайне разнообразны и имеют свои сильные и слабые стороны. Некоторые относят объекты к определенному классу, другие предсказывают числовые значения. Существуют и алгоритмы, определяющие сходства и различия допускающих сравнение сущностей (например, людей, машин, процессов, кошек, собак). При этом все алгоритмы обучаются на примерах (опыте) и умеют применять полученные знания к новым, ранее не встречавшимся случаям, то есть способны к обобщению.

На конкурсе программы «Собаки против кошек» на этапе обучения осуществлялось корректное выполнение классификации, используя множество алгоритмов. На каждой из миллионов обучающих итераций программа производила классификацию, измеряла полученный результат и затем по необходимости корректировала процесс в поисках постепенного улучшения. Лучшим результатом корректного распознавания у машин составило 98,914% тестовых изображений, что значительно превышает возможности человеческого распознавания, которое составляет примерно 7%. Процедура показана на рисунке 1.1.

Процесс машинного обучения работает по принципу анализа изображения с метками и построения модели, которая в свою очередь используется процессом воспроизведения (предсказания) для классификации новых изображений.

На примере подтверждается как одно изображение с кошкой было распознано некорректно.

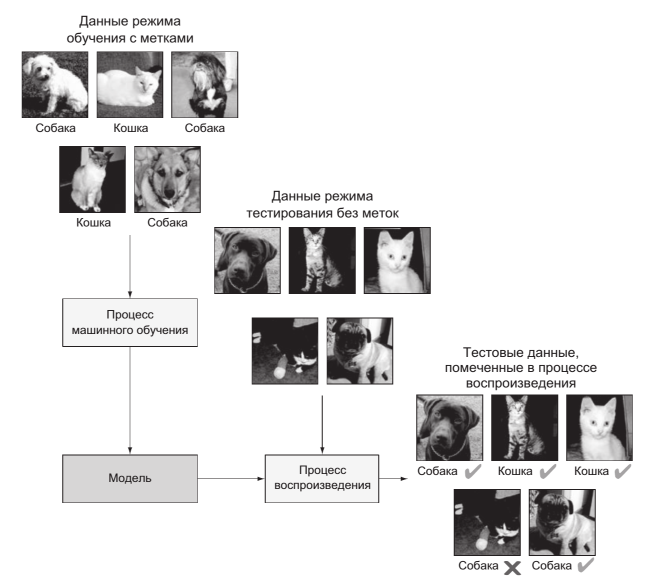


Рисунок 1.1 – Процесс машинного обучения для алгоритма,

отличающего кошку от собаки

Процесс обучения на основе данных и последующего применения полученных знаний для обоснования будущих решений – чрезвычайно мощный инструмент. Машинное обучение быстро превращается в двигатель современной экономики, управляемой данными.

Аналогия между человеческим и машинным обучением закономерно заставляет вспомнить такое явление, как искусственный интеллект.

Машинное обучение – это одна из форм искусственного интеллекта, так как ИИ представляет собой обширную область, включающую робототехнику, обработку лингвистической информации и системы машинного зрения.

В таблице 1.1 перечислены широко распространенные техники машинного обучения с учителем и варианты их практического применения.

Таблица 1.1 – Варианты применения машинного обучения с учителем, систематизированные по типам задач

|  |  |  |
| --- | --- | --- |
| Задача | Описание | Пример применения |
| Классификация | На основе данных определяется дискретный класс для каждого объекта | Фильтрация спама, анализ тональности текстов, обнаружение мошенничества, рассылка целевых рекламных объявлений, прогнозирование оттока клиентов, обработка заявок на техническую поддержку, персонализация контента, выявление производственных дефектов, сегментация потребителей, обнаружение событий, изучение геномов и эффективности лекарственных средств. |
| Регрессия | На основе данных предсказывается фактическое значение параметра | Прогнозы на рынке ценных бумаг, прогноз спроса, прогноз цены, оптимизация аукциона реальных объявлений, управление рисками, управление активами, прогнозы погоды, спортивные предсказания. |
| Рекомендация | Предсказывается альтернатива, которую предпочтет пользователь | Предложения продуктов, подбор персонала, конкурс Netflix Prize, онлайн знакомства, предложение контента. |
| Заполнение  Пропусков | Вывод значений отсутствующих входных данных | Неполные истории болезни, отсутствующая информация о клиентах, данные переписей. |

Процесс получения знаний человеком превосходит своей сложностью самые совершенные алгоритмы машинного обучения, но у компьютера есть преимущество в виде большей емкости для запоминания, извлечения и обработки данных. Накапливаемый опыт представлен в форме данных за длительный период времени, обработанных с помощью описанных методов, которые позволяют получать и оптимизировать алгоритмы, реализующие способность к обобщениям [4].

### **1.2.2 Принятие решений на основе данных**

В процессе работы группы иностранных ученых проведена экономическая задача, которая упрощается с помощью машинного обучения и показывает альтернативное преимущество подхода МО. На примере организации, которая предоставляет кредиты физическим лицам для открытия малого бизнеса в неблагоприятных районах. Первоначально организация получает несколько заявлений в неделю, что позволяет вручную прочитать каждое из них и в последующем принять решение об одобрении кредита. Этот процесс схематично представлен на рисунке 1.2.

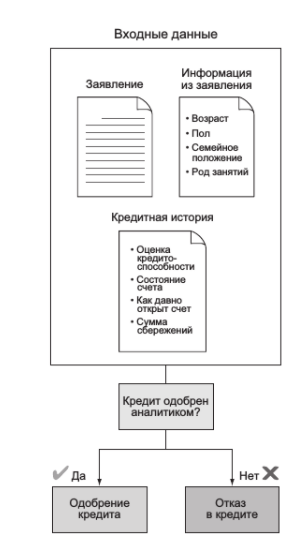


Рисунок 1.2 – Схема одобрения кредита

По мере расширения спроса на кредиты организации количество заявлений увеличивается в десятки раз в неделю, что приводит к чрезмерному увеличению рабочего дня сотрудников. Несмотря на сверхурочную работу, организация не справляется с наплывом клиентов, которые в процессе длительного ожидания ответа организации отказываются от услуг и отправляются в конкурирующую организацию. Очевидно, что обрабатывание заявлений вручную становится нерационально, что приводит к сильному стрессу сотрудников и качеству своевременной работы организации.

Возникает вопрос перед кредитной организацией по оптимизации решения данной проблемы. По опыту прошлых лет организации применяют традиционные способы позволяющие ускорить процесс анализа заявлений.

Одним из традиционных способов является анализ вручную, что подразумевает увеличение штата сотрудников.

Необходимо для работы с заявлениями нанять еще одного аналитика. Найм нового сотрудника приведет к дополнительным финансовым расходам предприятия, но улучшит своевременность обработки заявок за неделю.

По истечению определенного времени и увеличения количества заявок в предприятии до 1000 в неделю возникает потребность в расширении штата аналитиков в два раза. При проведении анализа ситуации напрашивается вывод, что в долгосрочной перспективе такая методика не рентабельна: доход от новых заемщиков пойдет на зарплату новым сотрудникам, а не в фонд кредитования и прибыль организации. Увеличение штата по мере роста спроса препятствует развитию бизнеса. Бесконечное увеличение штата аналитиков непосредственно повлияет на качество отбора сотрудников и соответственно на эффективность работы организации. Более того, сам процесс найма – длителен во времени и не выгоден финансово, так как требует дополнительное обучение новых сотрудников.

Следующим традиционным способом выступает бизнес–правила. Основным параметром при рассмотрении данного способа является анализ своевременного погашения полученных кредитов гражданами. На примере полученных за несколько месяцев работы организации из 2500 заявлений на кредит одобрена 1000. Из них 700 заявителей погасили кредит вовремя, а остальные 300 просрочили погашение. Эта ситуация представлена на рисунке 1.3. Этот исходный набор данных крайне важен для автоматизации процесса одобрения кредита.



Рисунок 1.3 – Схема одобрения и погашения кредита

При применении данного способа осуществляется поиск связей между данными заявителя и количеством погашенных кредитов. Применяется набор правил фильтрации, которые на выходе дают положительный результат по своевременному погашению кредитов. Ручной анализ поступающих сотен заявок, приводит к огромному опыту сотрудника и позволяет отличить потенциального кредитора от некредитоспособного. В процессе проведения проверки данных о погашениях кредитов обнаруживаются определенные тенденции:

* большинство заемщиков с лимитом кредитования более $7500 не выполняют своих обязательств по кредиту;
* большинство заемщиков, не имеющих контокоррентного счета, погашают кредит в срок.

В последующем на основании полученного опыта и анализа возможна проектирование механизма фильтрации, который сокращает количество заявлений, вручную проверенных по двум вышеуказанным критериям. Первый фильтр будет автоматически отказывать всем заемщикам с лимитом кредитования более $7500, так как накопленные данные показывают, что 44 из 86 заемщиков, взявших кредит, превышающий $7500, нарушают дату погашения. Приблизительно 51% из потенциально максимально возможных кредиторов не выполняют обязательств по отношению остальным 28%. Данный фильтр является эффективным способом отсечения заемщиков с высоким риском невозврата, но следует учитывать количество крупных кредиторов, которых всего 8,6% (86 из 1000) из одобренных заявок. Следовательно, более 90% анкет необходимо обрабатывать вручную, что требует создания дополнительной фильтрации.

С созданием второго фильтра указанный способ автоматически принимает любого заявителя при отсутствии контокоррентного счета, так как погашение кредита заемщиками составляет З48 из З94 (88%). Применение этого фильтра приводит к увеличению количества автоматически принимаемых или отклоняемых заявлений до 45%. Соответственно, количество вручную обрабатываемых анкет составит более 50%. Фильтрация по двум бизнес–правилам позволяет уменьшить количество проверяемых вручную заявок до 52%. Рисунок 1.4 демонстрирует правила фильтрации в виде схемы.

Два способа бизнес–правила приводят к увеличению количества обрабатываемых заявок в два раза и без найма второго аналитика, так как вручную решение принимается только по 52% новых заемщиков. Кроме того, имея данные по 1000 заявок с известным результатом, возможно предвидеть, что механизм фильтрации будет ошибочно отклонять 42 заявки из каждой 1000 (4,2%) и ошибочно принимать 46 из каждой 1000 (4,6%).

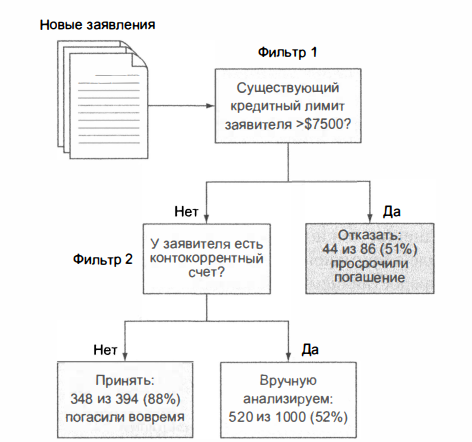


Рисунок 1.4 – Правила фильтрации заявок

По мере увеличения количества заемщиков возникает потребность в автоматическом приеме или отклонении заявок, без риска роста убытков из–за нарушения долговых обязательств. Дальнейшее дополнение новых бизнес–правил, приведет к следующим проблемам:

* по мере усложнения системы фильтрации возникнут осложнения в эффективности фильтров;
* бизнес–правила со сложной системой фильтрации приведет к нарушению отлаженной работы и сложности удаления устаревших и малозначительных правил;
* формирование новых правил не является статистически строгим, так как более тщательное изучение данных не гарантирует улучшения качества отбора;
* признаки возвращаемого кредита подвергаются временным изменениям, ввиду постепенной смены контингента заемщиков, и отсутствия адаптации системы. Эффективная работа системы требует постоянного редактирования.

Указанные проблемы сводятся к одному критическому недостатку в системе с применением бизнес–правила, это невозможность автоматического изучения предоставленных данных.

Системы, управляемые данными, от простых статистических моделей до более проработанных обучающихся рабочих процессов, позволяют избежать проблем такого рода [5].

## 1.3 Рабочий процесс: от данных до внедрения

Базовый процесс интеграции моделей машинного обучения в приложения или конвейеры данных распределены на пять шагов: подготовка данных, построение модели, оценка, оптимизация и прогноз на новых данных. Этапы следуют друг за другом в определенном порядке, но большинство реальных приложений с машинным обучением требует множественного повторения каждого этапа в процессе последовательных приближений. На рабочем процессе реальных систем с машинным обучением данные за прошедшие периоды позволяют построить модель на базе алгоритма МО. Следующим шагом является оценка производительности модели и оптимизация точности и масштабируемости в соответствии с поставленными требованиями. Конечная модель выдвигает прогнозы для новых данных.

Сбор и подготовка данных для систем с машинным обучением имеет представление в виде таблицы, не учитывая ранее имеющую иную форму. Схема рабочего процесса представлена на рисунке 1.5.

Данные в таблице распределены по строкам и столбцам, каждая строка соответствует изучаемому экземпляру (instance), а столбец – значению экземпляра. Несмотря на некоторые исключения, в большинстве случаев алгоритм принимает данные в вышеуказанном формате. В наборе данных, собранном в виде таблицы, строки являются экземплярами, а столбцы признаками.

На рисунке 1.6 проиллюстрирован обыкновенный набор данных, представленный в виде таблицы.



Рисунок 1.5 – Схема рабочего процесса

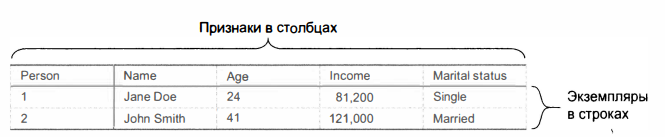


Рисунок 1.6 – Простой набор данных

На рисунке 1.6 столбцы в таблице содержат данные одного типа, а данные в строках принадлежат разным типам. На рисунок 1.6 представлены данные четырех типов: строковая переменная Name, целочисленная переменная Age, вещественная переменная Income и категориальная переменная Marital status (принимающая дискретное число значений). Набор данных представленные выше является гетерогенным (в отличие от гомогенного).

Реальные данные распределены хаотичными способами, на примере этапа сбора данных измерение какого–либо значения и возращение на предыдущий шаг для поиска недостающего фрагмента информации не представляется возможным. В подобных случаях одни из ячеек таблицы останутся пустыми, что усложняет построение модели и последующее прогнозирование. В некоторых случаях сбор данных осуществляется вручную, что приводит к ошибкам на этапе выполнения повторяющихся задач и часть сведений оказывается некорректной. Возникает необходимость умения работы с подобными сценариями и знание того, как конкретный алгоритм ведет себя при наличии недостоверных данных.

Следующим шагом является обучение модели на данных. Начальный этап построения успешной системы с машинным обучением является задачей, решение которой вытекает из данных. Таблица с личными данными, упомянутая выше формирует модель МО, предсказывающую семейное положение заявителя и информация такого рода необходима при выборе демонстрируемой пользователю рекламы.

В таблице, переменная Marital status использована как целевая (target) или как метка (label), а все оставшиеся переменные – признаки (features). На этом этапе алгоритм МО осуществляет набор входных признаков и успешно предсказывает значение целевой переменной. Заявители, которые на начальном этапе не указали семейное положение подвергаются использованию модели для его предсказания.

Рисунок 1.7 демонстрирует процесс на примере небольшого набора данных [6].

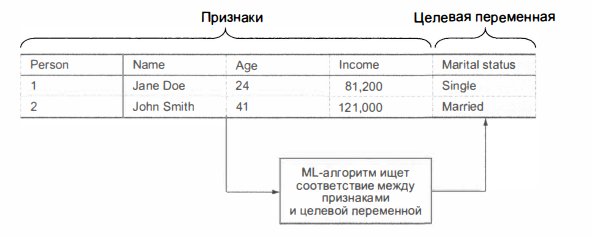


Рисунок 1.7 – Процесс моделирования с машинным обучением

На данном этапе алгоритм МО выполняет отображение входных признаков на выходные данные, но для построения работающей модели нужно более двух рядов. Одним из преимуществ алгоритмов машинного обучения в сравнении с другими распространёнными методами является умение обрабатывать множество признаков. Рисунок 1.7 демонстрирует только четыре признака, из которых идентификатор заявителя и его имя никак не помогут предсказать его семейное положение. Некоторые алгоритмы в достаточной степени невосприимчивы к неинформативным признакам, в то время как другие дают более точные предсказания, когда такие признаки убираются из рассмотрения [3].

Ценные сведения в некоторых случаях могут быть извлечены и из неинформативных на первый взгляд признаков. Например, такой признак, как местоположение, который дает информацию о плотности населения. В проектах МО вариант усовершенствования данных, называемый извлечением признаков (feature extraction), является одним из важных аспектов.

Используя на практике модель МО возможны прогнозы для новых данных с неизвестной целевой переменной, отображенные на рисунке 1.8.

Предсказанная целевая переменная возвращается в виде, в которой раннее фигурировала во взятых для обучения модели исходных данных.

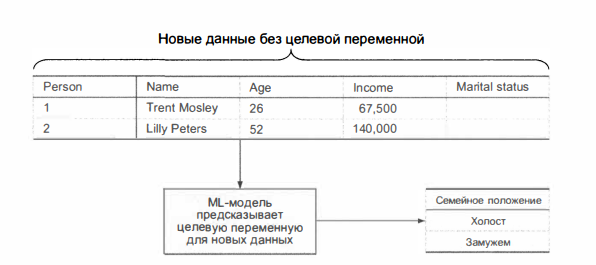


Рисунок 1.8 – Применение модели для предсказания новых данных

Прогнозирование с помощью модели является заполнением пустого столбца новыми значениями. Некоторые алгоритмы МО включают в результат работы, связанные с каждым классом вероятности.

В вышеупомянутом примере вероятностная модель МО, на выходе иллюстрирует нового заявителя с двумя значениями: вероятность нахождения в браке или нет.

Любая система, связанная с машинным обучением, занимается созданием моделей и их применением для получения прогнозов.

## 1.4 Усовершенствованные способы повышения эффективности

Существуют различные дополнительные техники, которые часто применяются для дальнейшего повышения производительности моделей. Для некоторых наборов данных и задач описанные техники позволяют значительно увеличить точность предсказаний, при этом теряя зачастую скорость работы на стадиях обучения и прогнозирования.

### **1.4.1 Предварительная обработка данных и проектирование признаков**

Предварительная обработка данных и проектирование признаков является неотъемлемой частью вышеупомянутой техники. Производительность модели повышается путем проведения очистки данных и извлечения из них дополнительных значений.

В любой предметной области требуются специальные знания, позволяющие решать, какие именно данные подлежат сбору. Именно эти знания применяются для извлечения из собранных данных ценной информации, которая добавляется к признакам строящейся модели. Данная работа называется процессом проектирования признаков (feature engineering). Массу времени занимает часть процедуры оптимизации, где процесс проектирования признаков осваивается с основными этапами рабочего процесса MО. На этапе процедуры оптимизации необходимо задействовать знания и воображение, с целью выявления способов усовершенствования модели путем исследования данных и извлечения из них скрытой информации.

Важными примерами проектирования признаков являются:

* дата и время. Эти переменные часто встречаются в наборах данных, при необходимости из них можно извлечь ценную информацию. Например, при выборе демонстрируемой рекламы, необходимым является наличие переменных: время, день недели и время года. Благодаря проектированию признаков эти сведения извлекаются из переменных даты и времени, и применяются в модели. Возникают такие переменные и при наблюдении за повторяющейся активностью, например за повторными визитами посетителя на сайт в течение месяца или года. В этом случае они позволяют вычислить промежутки между визитами, что может дать информацию для прогнозирования;
* местоположение. В некоторых наборах данных присутствуют координаты в виде широты и долготы или названия местностей. Указанная информация актуальна и при этом позволяет извлечь дополнительные параметры, необходимые для решения более специфических задач. Пример, при прогнозировании результатов выборов по стране требуется извлечение информации о плотности населения, среднем доходе и процентной доле бедных;
* цифровые средства коммуникации. В эту группу переменных, как правило входят следующие данные: тексты, документы, изображения и видео. Изображения первым делом дают информацию о контурах, формах и спектре цветов. С помощью математических преобразований они классифицируются, давая набор признаков, пригодных для алгоритмов классификации.

Рисунок 1.9 показывает интеграцию проектирования признаков в рабочий процесс машинного обучения [7].

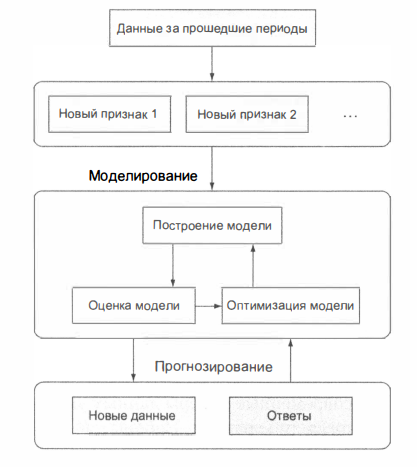


Рисунок 1.9 – Этап проектирования признаков, включенный в исходный рабочий процесс машинного обучения

### **1.4.2 Непрерывное совершенствование моделей**

Как правило, традиционные модели МО статистические и при необходимости перестраиваются, но во многих случаях данные и прогнозы будут возвращаться обратно в систему. Целесообразно, чтобы модель постепенно совершенствовалась и адаптировалась к изменениям в этих данных. Существуют алгоритмы МО, поддерживающие данный тип динамического обучения (online learning).

На рисунке 1.10 проиллюстрировано, каким образом непрерывное переучивание встраивается в рабочий процесс МО, путем возвращения прогнозов в модель с целью ее непрерывного улучшения.



Рисунок 1.10 – Диаграмма динамической системы МО

### **1.4.3 Масштабирование моделей**

Современные наборы данных увеличиваются в размерах крайне быстро. Наборы для методов МО с учителем, в которых целевые ответы входят в обучающую выборку, были традиционно небольшими, так как к получению ответов прибегали к помощи человека. В настоящее время множество данных продуцируются непосредственно измерительными элементами, машинами или компьютерами, и для обработки таких объемов требуются уже масштабируемые алгоритмы МО.

МО с учетом ранее рассмотренных моделей показывают свою практичность в применении и является наиболее точным, автоматизированным, быстрым, настраиваемым и масштабируемым, чем традиционные конструируемые вручную системы на базе бизнес–правил [8].

# 2 Основы программной инженерии

Основные компоненты CS–науки: Computer Engineering (компьютерная инженерия), System Engineering (системная инженерия), Software Engineering (SE – программная инженерия), соответствующая технологиям программирования.

Компьютерная инженерия – это теории, принципы и методы построения компьютеров (frameworks, суперкомпьютеров), а также системного ПО ЭВМ (ОС, трансляторов, загрузчиков).

Системная инженерия – это теория, методы и принципы построения информационных систем, систем управления и Computer System.

Программная инженерия – это система методов, способов и дисциплин планирования, разработки, эксплуатации и сопровождения ПО.

В научном труде В.М. Глушкова «Безбумажная информатика» рассмотрены принципы и методы построения информационных систем. Согласно этому труду, информационные системы – это компьютерные системы обработки информации на предприятиях, в органах управления и в производственной сфере. Базис таких систем – документы, не бумажные, а электронные, и система документооборота на всех уровнях управления государственных предприятий и органов.

Информационные технологии – базис компьютерной инфраструктуры современных корпораций, предприятий и государственных органов управления, на которых главным источником деятельности является информация и решение различных задач обработки информации локального и глобального характера [9].

## 2.1 Определение базовых понятий программной инженерии

В научной работе группы советских ученых «Проблематика программной инженерии» отображены точки зрения многих отечественных и зарубежных специалистов в области ПИ. Определены базовые понятия, смысл и методы создания ПП на начальном периоде развития ПИ.

Программная инженерия, впервые появилась в 1968 году и означало «систематический подход к разработке, эксплуатации, сопровождению и прекращению использования программных средств». С этим термином связаны важные разработки в области ТП и ПИ. Они публиковались в ряде научных зарубежных журналах: IEEE Transaction on Software engineering, Software Engineering Journal, Computer Technology, а также в советских журналах – Современные проблемы кибернетики, Программирование и другие. Для ядра SWEBOK была разработана международная программа обучения ПИ – Curricula 2001.

ПИ является инженерно-научной дисциплиной, методы, средства и инструменты которой обеспечивают качественный и производительней труд лиц, занимающихся различными видами деятельности по созданию ПО и использованию программных систем и ИС на инженерной основе. Появление ПИ знаменует переход от отдельного мануфактурного производства программ к промышленному. В нем для заданного предмета труда (программного объекта) определены: спецификация объекта (эскиз), действия (операции и процедуры), выполняемые исполнителями с целью создания по эскизу опытного образца ПС с заданными функциями и требуемым качеством. На основе опытного образца осуществляется его тиражирование для массового применения [10].

Появлению термина ПИ в СССР предшествовал термин технология программирования, который обозначал методы, средства и инструменты, обеспечивающие процесс создания ПС. Эти определения настолько близки, что фактически их можно трактовать как дальнейшее развитие ТП в плане обеспечения программистского труда инженерными методами организации (планирование, учёт, контроль) выполнения работ. Такое развитие, по существу, означало переход от одиночного создания программ отдельными лицами к промышленному их производству со всеми вытекающими проблемами организации труда.

На первом этапе создания сложных систем главную роль в процессе программирования играли интуиция и интеллектуальные способности отдельных программистов. Они вкладывали свои знания и опыт, творческие способности в методы и средства создания ПС для выполнения на ЭВМ, что приводило к некоторым ограничениям на свободу действия рядовых программистов в виде определенных регламентаций, правил и стандартов.

В результате подошли, как и в промышленности, к технологиям, при которых не только определяется предмет труда, выполняемый на операциях и процессах, но и проводится оценка этого труда на трудоемкость и стоимость, а также делается оценка показателей качества (надёжность, эффективность и другие).

В становлении ТП и ПИ сложилось три основных подхода:

* инженерный;
* управленческий;
* инженерный (программный).

Инженерный подход сравним с технологией промышленного производства, в которой рассматриваются регламентированные последовательности технологических операций, призванные гарантировать получение продукции с заданными техническими и качественными свойствами. ТО в свою очередь связаны с индивидуальными особенностями технических исполнителей, которые осуществлялись без учета их творческих способностей. По мнению некоторых специалистов в области ПИ настоящий подход воспрепятствует последующему прогрессу.

Несмотря на это, в сфере ПИ существуют виды работ, относящиеся к данному подходу: правильное копирование программ и документации; настройка и генерация программ; ввод и контроль данных и другие. Эти виды работ наиболее близки к технологическому производству. В нём также применяются специализированные работы (отладка, анализ, тестирование), которые могут выполняться по жестко отработанной, многократно проверенной технологии.

Вместе с тем данный подход позволяет творческие способности одних специалистов закладывать в более трудные процессы создания ПС, сделав труд разработчика программ более инженерным и производительным. Процесс разработки ПО и систем включает операции, требующие элементов творчества программистов, которые не ограничены определенными рамками процесса создания ПС, а именно: заданной последовательностью работ и результатом, зависящим от заданных в техническом задании требований к готовому программному продукту.

Управленческий подход основывается на четких организационных структурах группы разработчиков программ и требует обязательного выполнения нормативных документов и стандартов при разработке программ и систем. Данный подход к сожалению не учитывает основ ПИ, ориентированных на оптимизацию отношений руководителя с подчиненным и с потребителями в зависимости от специфики ПС. Инструментом оптимизации их отношений являются инструкции по созданию продукта и обеспечению его качества. Для управления качеством был разработан стандарт ГОСТ 2844–89 «Оценка качества программных средств». Он предусматривает большой объем работ по экспортированию отдельных свойств и характеристик ПС в продукт на этапах жизненного цикла. Оценка качества ПО проводилась по многим оценочным элементам, состоящих из 50 критериев и 6 факторов.

Однако экспертная оценка недостаточно точно отражала реальное качество ПО, и накладывала дополнительные расходы на его оценку. Методы управления разработкой продукта ПО, влияют на взаимные договоренности между заказчиком и исполнителем ПО, а также менеджером и программистами.

Инженерный (программный) подход основывается на автоматизации средств создания ПО и методов программирования.

Известны попытки свести решение различных проблем повышения производительности труда и качества разработки ПО к созданию или освоению технологических ПС простейшего или общего характера, автоматизирующих все программистские работы с помощью программных инструментов (software tools). Первоначально возникала опасная возможность замены профессионального мастерства исполнителей готовыми инструментами и средствами. При этом инструменты оказывались ненадёжными, трудоёмкими в использовании с имеющейся документацией.

Такие обстоятельства зачастую вели к игнорированию хороших сторон рассматриваемого подхода, связанных с методологическим характером программирования, включающим:

* разработку ПО и программной документации на него;
* комплексное управление качеством ПО;
* технику модульного и структурного программирования.

В практике программирования каждый из указанных подходов самостоятельно не использовался. Как правило, программист выбирает подход, подходящий к особенностям среды, в которой он работает. Программный объект является средством ПИ. Он обладает следующими признаками:

* многократной применяемостью;
* надёжной работоспособностью;
* спецификой, выгодно выделяющей данное средство от всех других, функционирования объекта разработки, его внедрения и эксплуатации.

В связи с изменчивостью условий ПО и среды функционирования объекта на ЭВМ первый признак недостижим без обеспечения соответствующего качества продукта, гарантирующего передачу его другим лицам и организациям.

Продуктом деятельности программиста может быть не только программа, но и технология ее изготовления, представленная методически или инструментально [10].

## 2.2 Основные объекты разработки программной инженерии

Основным понятием ПИ является объект разработки – программа во всех ее проявлениях на этапах жизненного цикла, а также программная система, методы, средства и инструменты их изготовления.

Программа – это объект разработки, который не является осязаемым (нельзя пощупать, взвесить) человеком, а доступен пониманию ЭВМ, для которой написан. Готовая программа – это программный продукт, реализующий определённые функции (задачи) некоторой предметной области, процесс проектирования и разработки которого осуществляется соответствующими методами, средствами и инструментами ПИ на технологических процессах.

Объектами разработки могут быть: модуль, программа, комплекс программ, пакет прикладных программ, система и другие. Объект либо сам является отдельной конструктивной единицей разработки, реализующей элементарную функцию ПрО, либо состоит из взаимосвязанного их набора.

Для объекта разработки первоначально формируется его модель, в которой специфицируются реализуемые функции и элементы данных, устанавливаются их связи и отношения, а также для объекта определяется его спецификация или прототип.

В техническом задании на разработку объекта, программы выдвигаются определенные требования к составу реализуемых функций и их характеристик – работоспособность (свойство надежности), быстродействие, удобство общения (свойство эргономичности), мобильность (свойство переносимости из одной среды в другую) и другие.

В зависимости от требований к условиям функционирования объекта (работать с большим быстродействием в условиях реального времени или с большой точностью в условиях космических и военных систем) требования к его свойствам отличаются и влияют на организацию процессов разработки систем.

С точки зрения теории автоматов любой объект имеет начальное (исходное), промежуточное и конечное состояния.

Начальное состояние – это исходная модель объекта (эскиз опытного образца) [11].

Промежуточное состояние – это изменённое состояние, отличное от начального и конечного состояний объекта, полученное на определенном этапе ЖЦ под воздействием соответствующих данному этапу программных методов и средств. Промежуточным состоянием объекта, например, является эскизный, технический и рабочий проекты.

Конечное состояние объекта – это программный продукт, готовый для исполнения требуемых от него функций. Конечным состоянием считается опытный образец, который передаётся в опытную эксплуатацию либо на тиражирование (производство).

В качестве аппарата задания начального состояния объекта используются языки спецификации (SADT, SSADM, PSA, PSI и так далее), а для задания промежуточных состояний используются языки программирования.

Метод разработки (программный метод) – это способ или планомерный подход к достижению той цели, которая ставится перед объектом разработки.

Наиболее распространенные методы проектирования и разработки: нисходящий, восходящий, модульный, метод расширения ядра, метод сборочного программирования. Метод определяет стратегию проектирования и разработки.

При нисходящем (сверху вниз) проектировании после определения требований к объекту формируется его функциональная и системная архитектуры, при которых декомпозированы все задачи ПрО и построена модель объекта. Задачи, в свою очередь, развиваются до понятий и функций объекта, выражаемых в базовых терминах рассматриваемой ПрО. Каждая функция объекта разрабатывается последовательным уточнением до определения элементов системной архитектуры.

Восходящий метод (снизу верх) является обратным нисходящему методу и начинается с определения элементарных базовых понятий ПрO и формирования из них более крупных понятий, приводящих в конечном итоге к определению некоторой функции ПрО. Для сформированной функции подбирается или вновь разрабатывается программный элемент (модуль, модель программы, макрос, заготовка). В этом плане данный метод является методом от готового. Развитием восходящего метода является метод прототипирования, при котором для объекта вначале создаётся «грубый» его прототип из готовых компонентов, от которых требуется, чтобы они соответствовали функциям объекта без учета эксплуатационных характеристик. Цель прототипирования состоит в том, чтобы отработать «каркас» объекта и его функциональные возможности, а затем улучшать характеристики свойств программных элементов.

Метод расширения ядра характеризуется начальным выделением множества вспомогательных функций. Наиболее эффективным методом выделения является метод, использующий анализ данных и определение модулей, обрабатывающих различные информационные структуры. Для этого может применяться метод Джексона, в основе которого лежит принцип соответствия организации программы этапам преобразования обрабатываемых данных.

Несмотря на различие этих методов в стратегиях проектирования, общее, что их объединяет – это модульное или блочное (программное) представление объекта разработки.

Метод модульного программирования обеспечивает декомпозицию задач ПрО на отдельные функции, вплоть до элементарных, каждой из них сопоставляется отдельная программа пли модуль. Каждый модуль должен обладать интерфейсом для его связи с другими модулями и программами. Применение данного метода предоставляет по сравнению с другими значительные преимущества в плане организации и управления разработкой, но вместе с тем требует создания определенных механизмов языкового и программного характера для решения проблем интерфейса при сборке модулей в более сложные программные структуры. Расширением данного метода является метод сборочного программирования, обеспечивающий интеграцию программных элементов различной степени сложности.

Таким образом, указанные методы проектирования и программирования взаимосвязаны, они используют друг друга. Так, при применении восходящего метода проектирования для систем обработки данных на ранних этапах ЖЦ программного объекта используются методы модульного программирования, расширения ядра и другие.

Технологический процесс – это взаимосвязанная последовательность операций, выполняемых при разработке объекта. Процесс предназначен для перевода объекта из одного состояния в другой соответствующими программными методами и средствами.

Со временем сформировался набор типовых функций автоматизируемых ПрО (СОД, АСНИ, САПР, АСУ и другие). Им соответствуют типовые ТП, которые вместе со специализированными образуют линию программ функционально–ориентированного типа для каждой системы.

Технологическая линия задает набор ТП разработки функций объекта, представленных совокупностью технологических операций, которые последовательно и систематически преобразуют состояния объектов, включая заключительное его состояние – готовый программный продукт. Особенность линии – отражение определенных функций ПрО, реализуемых в виде программ с заданными показателями качества.

Для одиночных программ количество операций в ТП меньше, чем для сложных. Независимо от сложности программного объекта основным условием выполнения процессов является их автоматизация. Процессы имеют модельное представление жизненного цикла объекта разработки.

Динамика изменения объекта может быть представлена в ТЛ последовательностью операций, переводящих объект из одного состояния в другое путем использования средств автоматизации операций процесса. Каждая конкретно разработанная ТЛ определяет множество допустимых состояний объектов, а на средства их автоматизации возлагается функция перевода и слежения за переходом в недопустимые состояния.

Инструмент – это программное, языковое или методическое средство, применяемое для получения состояния объекта в некотором законченном виде. В зависимости от этапа жизненного цикла и состояния объекта для работы с ним могут быть языки, трансляторы, генераторы.

Процесс преобразования модели объекта из одного состояния в другое не является полностью автоматизированным и имеется ряд CASE–систем, специально предназначенных для реализации конкретных ПрО.

Каждая инженерная дисциплина базируется на принципах и методах конструирования (разработки) и промышленного производства продуктов, которые затрагивают как организационные, так и управленческие аспекты производства. Основными вопросами управления разработкой объектов как инженерии ведения разработки являются:

* организация коллектива разработчиков (состав, структура, квалификация и другие);
* планирование работ, трудозатрат и обеспечение роста производительности труда;
* контроль хода разработки и оценка проектных решений в ходе разработки ПП;
* экономические вопросы (стоимость, ценообразование, стимулирование и другие);
* управление качеством.

Жизненный цикл – это совокупность последовательных процессов и всех действий (операций), связанных с превращением программы в продукцию производственно-технического назначения, начиная с анализа потребностей в автоматизации функций ПрО до создания ПС и кончая его моральным износом.

С ЖЦ связаны сроки и период разработки и эксплуатации ПС.

По срокам эксплуатации ПС разделяются на два класса:

* с краткосрочной эксплуатацией;
* с долгосрочной эксплуатацией.

К краткосрочным ПС относятся продукты научного творчества студентов, аспирантов, а также стажёров, занимающихся их созданием для проверки научной идеи. ЖЦ таких продуктов включает этапы проектирования и разработки. Этапу эксплуатации ПП соответствует процесс апробации созданного продукта.

Программным объектам второго класса соответствуют регламентированные процессы проектирования, разработки и эксплуатации. Они создаются в проектных и промышленных организациях, занимающихся реализацией на ЭВМ крупных народнохозяйственных задач. Объекты данного класса изменятся в диапазоне 100–110 тысяч команд, строк операторов ЯП со свойством изменяемости и модификации при их сопряжении и использовании. Такие объекты тиражируются и вместе с документацией передаются потребителю, отчуждаясь от разработчика. Срок жизни таких программ 10–20 лет, 70–80% этого срока приходится на эксплуатацию и сопровождение.

Программы с долгосрочным характером эксплуатации требуют применять в процессе разработки индустриальные методы организации и планирования разработки. На первое место выступают задачи экономического характера, способствующие повышению производительности, качества ПС и уменьшению сроков разработки. Главная задача – оценивание затрат на разработку ПС и установление конечной стоимости ПП.

ТЛ ориентированы на производства ПП и требуют технико-экономического обоснования процессов разработки, достижения высокого качества ПС, методов оценивания стоимости ПС и учета человеческого фактора в процессе разработки ПС [12].

Предложенная Б.У. Бoeмом модель стоимости (COCOMO) позволяет провести оценку стоимости и затрат на этапах ЖЦ с учётом ряда стоимостных атрибутов, показателей квалификации исполнителей и степени использования современных методов программирования:

* соотношение текущих и будущих расходов на изготовление ПП;
* эффективность ПС;
* методы расчета доходов и чистой стоимости;
* способы оценивания стоимости, основанные на планировании ресурсов, уточнения требований, учета плановых сроков и другие;
* методы экспертных оценок свойств ПП, рассматриваемых как факторы (надежность, сложность, быстродействие и другие), влияющие на стоимость;
* средства создания ПС (стили и методы программирования и инструменты).

На основе рассмотренного материала следует отметить, что ПИ включает в себя комплекс систем планирования (методов, способов и дисциплин), разработки, эксплуатации и сопровождения ПО. Она охватывает все этапы создания ПО от начала формулирования требований, разработки, сопровождения и окончательного его списания. Теоретическая основа ПИ базируется на теориях алгоритмов и программирования, методов вычислений и распределения коммуникаций. Практическое применение ПП основывается на теориях менеджмента, планирования процессов и необходимых для этого ресурсов, верификации и тестирования, оценивания рисков и качества. Технология создания ПП развиваются в направлении совершенствования программных и информационных технологий [12].

# 3 Машинное обучение в программной инженерии

Последние достижения в области машинного обучения вызвали широкий интерес в секторе информационных технологий к интеграции возможностей искусственного интеллекта в программной инженерии.

Почти десятилетие проходит без разрушительных изменений в доминирующей области применения индустрии программного обеспечения. Каждая смена приносит с собой новые цели разработки программного обеспечения, которые побуждают организации, занимающиеся разработкой программного обеспечения, совершенствовать свои методы разработки, чтобы учитывать новые аспекты предметной области.

Последняя тенденция, поразившая индустрию программного обеспечения, связана с интеграцией возможностей искусственного интеллекта, основанных на достижения в области машинного обучения. ИИ в широком смысле включает в себя технологии для рассуждения, решения проблем, планирования и обучения, среди прочего. Машинное обучение относится к статистическому моделированию [13].

В последние годы исследованием данного научного направления заинтересовались различные мировые научные школы и организации. Впервые группа ученых во главе с Салимом Амерши и Эндрю Бегела осуществила практическое исследование МО в ПИ на базе компании Microsoft. На начальном этапе работы группа ученых проводило наблюдение за командами разработчиков программного обеспечения в Microsoft, которые разрабатывали приложения на основе искусственного интеллекта. Команды разработчиков Microsoft использовали машинное обучение для создания наборов приложений, таких как Bing, поиска или виртуального помощника Cortana, а также такие платформы, как Microsoft Translator для перевода текста, голоса и видео в режиме реального времени, когнитивные сервисы для зрения, речи и понимания языка для создания интерактивных диалоговых агентов и платформа искусственного интеллекта Azure, позволяющая клиентам создавать свои собственные приложения для машинного обучения. Чтобы создать эти программные продукты, Microsoft использовала свои ранее существовавшие возможности в области искусственного интеллекта и разработала новые области знаний по всей компании. В этой работе описывается исследование, в ходе которого стало известно, как различные команды разработчиков программного обеспечения Microsoft создают программные приложения с функциями искусственного интеллекта, ориентированными на клиента. Для этого корпорация Microsoft интегрировала существующие процессы разработки гибкого программного обеспечения.

Команды разработчиков в ходе работ провели девятиэтапный рабочий процесс, основанный на предыдущем опыте разработки приложений искусственного интеллекта (например, поиск и NLP) и инструментов анализа данных (например, диагностика приложений и устранение ошибок отчетность). Группой ученых было выявлено, что различные команды разработчиков Microsoft объединили девятиэтапный рабочий процесс в уже существующие, хорошо развитые, гибкие процессы разработки программного обеспечения, предоставляя информацию о нескольких важных инженерных проблемах, с которыми могут столкнуться организации при создании крупномасштабных решений ИИ для рынка, а также осуществлен сбор рекомендаций Microsoft teams по решению этих проблем.

Определено три аспекта в области искусственного интеллекта, которые принципиально отличают ее от предыдущих областей прикладного программного обеспечения:

* обнаружение и управление версиями данных, необходимые для приложений машинного обучения намного сложнее, чем другие виды разработки программного обеспечения;
* настройка модели и повторное использование модели требуют совсем других навыков, чем обычно встречаются в командах разработчиков программного обеспечения;
* компоненты искусственного интеллекта сложнее обрабатывать как отдельные модули, чем традиционные программные компоненты. Модели могут быть сложными и испытывать немонотонное поведение с ошибками.

В исследовании группы ученых был проведен опрос сотрудников Microsoft о том, как они справлялись с растущими проблемами повседневной разработки ПО, специфичными для ИИ, а также с более крупными и существенными проблемами, присущими разработке крупномасштабной инфраструктуры и приложений ИИ.

Поскольку команды по всей компании имеют разный опыт работы с ИИ, было замечено, что отдельные проблемы присущие взглядам новых команд в процессе дальнейшей их работы теряли свою актуальность, хотя некоторые проблемы оставались важными для практики крупномасштабного ИИ [14].

Исследователи предприняли попытку создать показатель зрелости процесса, чтобы помочь командам определить, как далеко они продвинулись на своем пути к созданию приложений с искусственным интеллектом.

В качестве ключевого вывода анализа ученными обнаружено три фундаментальных отличия в создании приложений и платформ для обучения и внедрения моделей машинного обучения по отношению предыдущих областей применения. Во-первых, машинное обучение – это все, что связано с данными. Количество усилий и тщательности, необходимых для обнаружения, получения исходных данных, управления ими и создания версий, по своей сути более сложные и отличаются от выполнения того же самого с программным кодом. Во-вторых, создание настраиваемых и расширяемых моделей требует от команд не только навыков разработки программного обеспечения, но и почти всегда требуются достаточно глубокие знания в области машинного обучения, чтобы создавать, оценивать и настраивать модели с нуля. В-третьих, может быть сложнее поддерживать строгие границы модулей между компонентами машинного обучения, чем для модулей разработки программного обеспечения. Модели машинного обучения могут быть сложными взаимосвязанными способами, которые заставляют их влиять друг на друга во время обучения и настройки.

Уроки, которые извлекли из исследований различных команд Microsoft, адаптировавших свои процессы и методы разработки программного обеспечения для интеграции машинного обучения, могут помочь другим разработчикам ПО, встать на собственный путь создания приложений и платформ для искусственного интеллекта [15].

## 3.1 Процессы разработки программного обеспечения

Меняющиеся тенденции в области приложений в индустрии программного обеспечения повлияли на эволюцию программных процессов, используемых командами Microsoft. По крайней мере, в течение полутора десятилетий многие команды использовали гибкие методы, основанные на обратной связи, для разработки своего программного обеспечения, потому что им нужно было быстро реагировать на меняющиеся потребности клиентов с помощью более быстрых циклов разработки. Гибкие методы были полезны для поддержки дальнейшей адаптации, например, самый последний сдвиг в реорганизации многочисленных методов работы команды вокруг DevOps, который лучше соответствовал потребностям создания и поддержки приложений и платформ облачных вычислений. Переход на DevOps произошел довольно быстро, потому что эти команды смогли использовать предыдущие возможности в непрерывной интеграции и сборе диагностических данных, что упрощает реализацию непрерывной доставки.

Изменения в процессах не только меняют повседневную практику разработки команды, но и влияют на роли, которые играют люди. 15 лет назад многие команды в Microsoft в значительной степени полагались на триады разработки, состоящие из менеджера программы (сбор требований и планирование), разработчика (программирование) и тестировщика (тестирование). Внедрение DevOps этими командами объединило роли разработчика и тестировщика и интегрировало функции ИТ, операций и диагностики в основную линию команд разработчиков программного обеспечения [16].

В последние годы команды расширили свои возможности по анализу поведения клиентских приложений на основе диагностики, определению приоритетов ошибок, оценке частоты отказов и пониманию регрессий производительности благодаря привлечению специалистов по обработке данных, которые помогли впервые интегрировать статистические и машинные рабочие процессы обучения в процессы разработки программного обеспечения. В некоторых командах разработчиков программного обеспечения работают опытные специалисты по обработке данных, которые “делают все”, но поскольку наука о данных нуждается в расширении, их роли специализируются в экспертов предметной области, которые глубоко разбираются в бизнес-проблемах, разработчиков моделей, которые разрабатывают прогнозирующие модели, и разработчиков платформ, которые создают облачную инфраструктуру.

## 3.2 Рабочий процесс МО

Один из часто используемых рабочих процессов машинного обучения в компании Microsoft был представлен в различных формах в промышленности и исследованиях. Он имеет общие черты с предыдущими рабочими процессами, определенными в контексте data science и поиска данных.

Несмотря на незначительные различия, эти представления имеют общую сущность процесса, ориентированного на данные и множественные контуры обратной связи между различными этапами.

На рисунке 3.1 показано упрощенный вид рабочего процесса, состоящего из девяти этапов.

Некоторые этапы ориентированы на данные (например, сбор, очистка и маркировка), другие ориентированы на модели (например, требования к модели, разработка функций, обучение, оценка, развертывание и мониторинг).

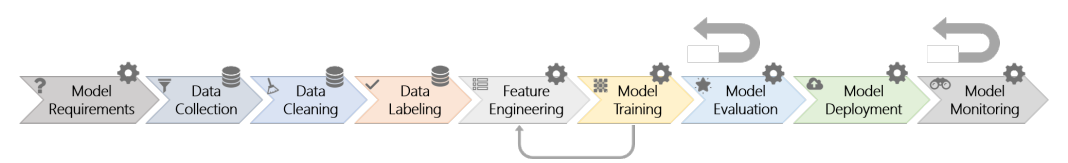


Рисунок 3.1 – Девять этапов рабочего процесса машинного обучения

В рабочем процессе существует множество контуров обратной связи. Большие стрелки обратной связи указывают на то, что оценка и мониторинг модели могут возвращаться к любому из предыдущих этапов. Меньшая стрелка обратной связи показывает, что обучение модели может вернуться к разработке характеристик (например, при обучении представлению).

На этапе требований к модели разработчики решают, какие функции возможно реализовать с помощью машинного обучения, а какие могут быть полезны для данного существующего или нового продукта. Самое главное, что на этом этапе они также решают, какие типы моделей наиболее подходят для данной проблемы. Во время сбора данных команды ищут и интегрируют доступные наборы данных (например, внутренние или с открытым исходным кодом) или собирают свои собственные. Часто они могут обучить частичную модель, используя доступные общие наборы данных (например, ImageNet для обнаружения объектов), а затем использовать трансфертное обучение вместе с специализированными данными для обучения более конкретной модели (например, обнаружение пешеходов). Очистка данных включает в себя удаление неточных или зашумленных записей из набора данных, что является общим делом для всех видов data science.

При маркировке данных каждой записи присваиваются метки, соответствующие истине. Например, у инженера может быть набор изображений, которые еще не были помечены объектами, присутствующими на изображении. Большинство методов контролируемого обучения требуют наличия меток для создания модели. Другие методы (например, обучение с подкреплением) используют демонстрационные данные или вознаграждения среды для корректировки своих правил. Метки могут быть предоставлены самими инженерами, экспертами в предметной области, или сотрудниками краудсорсинговыми онлайн–платформами.

Инженерия признаков представляет собой действия, которые выполняются для извлечения и отбора информативных признаков для моделей машинного обучения. Для некоторых моделей (например, сверточных нейронных сетей) этот этап менее явный и часто смешивается со следующим этапом – обучением модели.

Во время обучения модели, выбранные модели (использующие выбранные признаки) обучаются и настраиваются на основе чистых, собранных данных и их соответствующих метках.

Затем на этапе оценки модели инженеры оценивают выходную модель на протестированных или защищенных наборах данных с использованием заданных метрик. Для критически важных областей этот этап может осуществляться на основе обширной оценки человека. Код вывода модели развертывается на целевом устройстве (устройствах) и постоянно отслеживается на предмет возможных ошибок во время практического выполнения [17].

Для простоты вид на рисунке 3.1 является линейным, однако рабочие процессы МО нелинейны и содержат несколько контуров обратной связи. Например, если инженеры выявят между обучающими и реальными данными большой сдвиг в распределении, тогда они вернуться назад, чтобы собрать более репрезентативные данные и повторно запустить рабочий процесс.

Аналогичным образом, они могут пересмотреть выбор моделирования, сделанный на первом этапе, в случае изменения проблемы или изобретения более совершенных алгоритмов. В то время как циклы обратной связи типичны для процессов программного обеспечения, особенность рабочего процесса МО связана с количеством экспериментов, необходимых для получения хорошей модели в решении проблемы. Повседневная работа инженера, занимающегося машинным обучением, включает в себя частые итерации над выбранной моделью, гиперпараметрами и уточнением набора данных. Аналогичные экспериментальные свойства наблюдались в прошлом в научном программном обеспечении и совместном проектировании аппаратного и программного обеспечения. Этот рабочий процесс может стать еще более сложным, если система является интегративной, содержащей множество компонентов MО, которые взаимодействуют друг с другом сложными и неожиданными способами [17].

## 3.3 Программная инженерия для машинного обучения

Необходимость корректировки методов разработки программного обеспечения в последнее время обсуждается в контексте скрытого технического долга и устранения проблем интегративного ИИ. В этой работе определены различные аспекты архитектуры и требования системы машинного обучения, которые необходимо учитывать при проектировании системы.

Некоторые из этих аспектов включают скрытые контуры обратной связи, запутанность компонентов и размытые границы, немонотонное распространение ошибок, непрерывные состояния качества и несоответствия между реальным миром и наборами оценок. В связи с этим также обсуждается влияние использования программного обеспечения на основе МО на риск и безопасность стандартов ISO.

За последние пять лет в промышленности было предпринято множество попыток автоматизировать этот процесс путем создания фреймворков и среды для поддержки рабочего процесса МО и его экспериментального характера.

Однако текущие исследования и опросы показывают, что инженерам все еще сложно операционализировать и стандартизировать рабочие процессы.

Цель этой работы является выявление подробных сведений о положительных практических рекомендациях, относящихся к МО, используемых разработчиками в компании Microsoft.

### **3.3.1 Зрелость процесса**

Инженеры–программисты сталкиваются с постоянно меняющимся набором платформ и технологий, которые они должны изучить, чтобы создавать новейшие приложения для рынка программного обеспечения. Некоторые инженеры изучают новые методы и технологии в учебных заведениях и в последующем эти знания используют в своих организациях. Другие осваивают новые навыки на работе или на стороне, поскольку они предвидят потребность своей организации в скрытых талантах. Команды программистов, состоящие из отдельных инженеров с различным уровнем опыта в области навыков, необходимых для профессионального создания компонентов МО и инфраструктуры их поддержки, демонстрируют разный уровень владения навыками в зависимости от совокупности их опыта в данной области.

Дисциплина ПИ давно рассматривает совершенствование процессов ПО как одну из своих жизненно важных функций. Исследователи и практики в этой области разработали семь известных метрик для его оценки, включая модель зрелости возможностей. Модель зрелости оценивает программные процессы организаций на пяти уровнях: начальный (специальные процессы), повторяемый, определенный, работоспособный (количественно измеряемый) и эффективный (целенаправленное улучшение процесса). Воспользовавшись этой моделью, была создана первая модель зрелости для команд, создающих системы и платформы, в которые интегрированы компоненты МО.

*Интервью для исследования.*

В настоящее время практика работы по созданию и внедрению МО в ПО и сервисы находятся на начальной стадии развития и не является единой для всех групп разработчиков. Отсутствует тема внедрения систематического способа выявления ключевых заинтересованных сторон. Исходя из этого в работе использована стратегия выборки по принципу снежного кома, начав с руководителей команд с развитым использованием МО (например, Bing), руководителей команд, где ИИ является основным аспектом пользовательского опыта (например, Cortana), и сотрудников ведущих компанию по широкому внутреннему обучению в области ИИ и МО. При выборе информантов, группа ученых сформировала разнообразные команды с целью получения различного уровня опыта и части экосистемы (продукты с компонентами ИИ, фреймворки и платформы ИИ, ИИ, созданный для внешних компаний). В общей сложности было опрощено 14 инженеров–программистов, занимающих руководящие должности. Результаты представлены на рисунок 3.2. Интервью были полуструктурированными и специализированными в зависимости от роли каждого информанта. Например, при интервью с информантом 3 задавались вопросы, связанные с работой по надзору за командами, создающими архитектурные компоненты продукта.

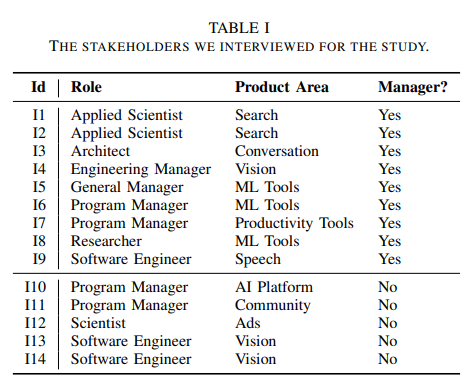


Рисунок 3.2 – Структура интервью исследования

### **3.3.2 Исследование**

На основе результатов интервью, была разработана анкета с открытым ответом, в которой основное внимание уделялось существующей практической работе, ее проблемам и передовым разработкам. О проблемах опрашивалось прямо и косвенно с целью выявления недостатков и подачи предложений по улучшению практической работы. Анкета была разослана 4195 ранее выбранным участникам по темам ИИ и МО. Результаты получены от 551 инженер–программистов, что составило 13,6% ответов. Для каждого пункта с открытым ответом от двух до четырех исследователей анализировали ответы с помощью сортировки карточек. Результаты сортировки карточек были проанализированы на предмет четкости и последовательности.

Респонденты были распределены по всем подразделениям компании и представляли разные должности, из них 21% руководители и 79% сотрудники, что помогло сбалансировать точку зрения большинства руководителей в интервью [18].

## 3.4 Применение искусственного интеллекта

Многие команды Microsoft расширили свои приложения с помощью МО. Проведен опрос респондентов на предмет использования ИИ в своих командах. Эти данные были дважды отсортированы: первый раз, с целью охвата области применения ИИ, второй раз, чтобы посмотреть алгоритмы MО, используемые для создания этих приложений.

В результате выявлено, что ИИ используется в традиционных (поиск, реклама, машинный перевод, прогнозирование покупок клиентов, распознавание голоса и изображений) и новых областях (выявление потенциальных клиентов, предоставление рекомендаций по оформлению презентаций и текстовых документов, предоставление уникальных возможностей рисования, здравоохранение и улучшение игрового процесса). Кроме того, МО активно применяется в инфраструктурных проектах для управления отчетами об инцидентах, определения наиболее вероятных причин ошибок, отслеживания мошеннической фискальной активности и мониторинга сетевых потоков на предмет нарушения безопасности.

Респонденты использовали широкий спектр подходов MО для создания приложений, от классификации, кластеризации, динамического программирования и статистики до моделирования поведения пользователей, анализа социальных сетей и совместной фильтрации. Организационные структуры компании специализировались по направлению деятельности. Например, отдел поиска активно работал с алгоритмами ранжирования и релевантности наряду с пониманием запросов. Другие подразделения компании работают над обработкой естественного языка, разрабатывая инструменты для распознавания сущностей, анализа настроений, предсказания намерений, обобщения, машинного перевода, построения онтологий, сходства текстов и связывания ответов с вопросами. Финансовый отдел и отдел продаж стремятся строить модели прогнозирования рисков и заниматься прогнозированием. Внутренние ресурсные организации используют алгоритмы оптимизации решений, такие как оптимизация ресурсов, планирование, ценообразование, проведение торгов и оптимизация процессов.

Итогом данного раздела считается, что интеграция компонентов МО используется по всем подразделениям компании. Важным фактором определения широкого диапазона применения компонентов МО является выявление точек зрения респондентов компании по результатам проведенных опросов [19].

# 4 Практики машинного обучения в программной инженерии

В этом разделе представлены точки зрения респондентов по основным проблемам, связанным с созданием крупномасштабных приложений и платформ MО и способы их решения. Классифицировали проблемы по категориям разделяя вопросы интервью и опросов по ответам, использовали предложения конкретных разработчиков ПО и исследователей ИИ для применения в практике ИИ командой разработчиков ПО.

По мере развития компонентов МО, их зрелости и интегрированности в более крупные программные системы, признана необходимость интеграции поддержки разработки MО в традиционную инфраструктуру разработки ПО. Отмечено важность автоматизации в безупречном опыте разработки, охватывающий все рабочие этапы согласно рисунку 3.1. Однако достижение такого уровня интеграции является сложной задачей ввиду различных характеристик модулей MО по сравнению с традиционными программными компонентами. Опыт предыдущих работ в этой области показал, что вариации неопределенности (ошибок), присущие алгоритмам обучения и управляемым данными, сложность компонентов, вызванная скрытыми контурами обратной связи, вносят существенные изменения (даже на отдельных этапах), которые ранее были хорошо изучены в ПИ (например, спецификация, тестирование, отладка) [19].

Несмотря на трудности, благодаря экспериментальному и итеративному характеру разработки MО, унификация и автоматизация повседневного рабочего процесса инженеров–программистов сокращает накладные расходы и способствует прогрессу в этой области.

Респонденты сообщают, что в процессе работы используют внутреннюю инфраструктуру компании (например, AEther) или создают конвейеры, специализированные для их частных случаев применения. Важно разработать надежный конвейер данных, способный непрерывно загружать и обрабатывать данные, позволяя инженерам опробовать множество вариантов алгоритмов ИИ с различными гиперпараметрами. Созданные этими командами конвейеры автоматизированы, поддерживают обучение, развертывание и интеграцию моделей с продуктом, частью которого они являются. Кроме этого, инженерами отмечено полезность значения «информационные панели» для пользователей.

Отдельные респонденты разрабатывают общедоступные IDE, позволяющие клиентам Microsoft создавать и внедрять свои модели (например, Azure ML для Visual Studio Code и Azure ML Studio). Целью этих разработок является оказание технической помощи инженерам в обнаружении, сборке, обработке, анализе и преобразовании данных, и в обучении, развертывании и поддержке модели. Также этот продукт создает условия для упрощенного его применения инженерами различной квалификации [19].

## 4.1 Доступность, сбор, очистка и управление данными

Поскольку многие методы МО основаны на обучении больших массивов данных, успех проектов, ориентированных на МО, в значительной степени зависит от доступности качества и управления данными. Маркировка наборов их данных требует больших затрат и времени, поэтому важно их доступность для использования внутри компании (с учетом ограничений соответствия требованиям). Респонденты уделяют особое внимание поддержке следующих атрибутов данных: доступность, точность, авторитетность, свежесть, латентность, структурированность, онтологическая типизация и связность. Автоматизация является жизненно важной сквозной задачей, позволяющей командам более эффективно агрегировать данные, извлекать функции, синтезировать помеченные примеры.

Обнаружено, что команды Microsoft сочли необходимым объединить инструменты управления данными со своими фреймворками MО, чтобы избегать фрагментации действий по управлению данными и моделями. Фундаментальным аспектом управления данными для МО быстрое развитие источников данных. Непрерывные изменения в данных могут происходит в результате операций, инициированных инженерами, и при поступлении новых сведений (например, данными датчиков, взаимодействиями с пользователем). Требуются методы управления и обмена данными, например, «Каждая модель помечена меткой происхождения, которая объясняет, на каких данных и версии модели она была обучена. Каждый набор данных помечен информацией, откуда получен, и какая версия кода использовалась для его извлечения (и любых связанных с ним функций).» Эта практика используется для сопоставления наборов данных с развернутыми моделями или для облегчения обмена данными и возможности их повторного использования [20].

## 4.2 Образование и профессиональная подготовка

В настоящее время интеграция МО активно продолжает распространение в продуктах, ориентированных на потребителя, например, компоненты МО широко используются в ПО для производственной деятельности (электронная почта, обработка текстов) и во встроенных устройствах (периферийные вычисления). Таким образом, инженеры с традиционным опытом разработки ПО должны научиться работать наряду со специалистами по MО. Многие сотрудники компании Microsoft сочли необходимым развитие образования инженеров различными способами. Во-первых, компания дважды в год проводит внутреннюю конференцию по МО и науке о данных, один день которой посвящен введению к основам технологий, алгоритмов и лучших практик. На конференции сотрудники докладывают о внутренних инструментах и технических деталях, лежащих в основе новых проектов и функций продуктов, а исследователи представляют свой вклад в передовые научные достижения. Во-вторых, ряд команд компании Microsoft проводят еженедельные открытые форумы по МО позволяя разработчикам расширить знания в области ИИ.

Наряду с результатами проведенных мероприятий осуществляются рассылки и онлайн–форумы с тысячами участниками, что позволяет каждому обсуждать технические вопросы об ИИ и МО, и освещать последние результаты научных конференций.

Отладка компонентов, формированных на основе данных, направлена не только на исправление ошибок программирования, но и на устранение проблем, возникающих из–за технических ошибок и не совершенности модели. По мере усложнения алгоритмов МО и методов оптимизации требуется активное исследования в области информационной достоверности моделей [20].

## 4.3 Эволюция, оценка и внедрение модели

Программное обеспечение, ориентированное на МО, часто пересматривается, что инициируется изменениями модели, настройкой параметров и обновлением данных, сочетание которых оказывает значительное влияние на производительность системы. Ряд команд сочли важным использовать строгие и гибкие методы для оценки своих экспериментов. Они разработали систематические процессы, применяя методы комбинированного облета (облет комбинации изменений и обновлений), включая множество показателей в карты оценки экспериментов и проводя оценку с участием человека для более чувствительных категорий данных. Команда одного из респондентов использует «карты для оценки и хранении информации о полетах». Автоматизация тестов так же важна в МО, как и в разработке ПО; командами были продуманы наборы тестов, которые отражают поведение модели. По мнению одного из респондентов: «Проводится выборочная оценка, а также человеку поручается анализ категории ошибок и вырабатывается гипотеза по выявлению источника проблемы».

Быстро меняющиеся итерации модели требуют более частого развертывания. С целью гарантированного и качественного развертывания системы инженеры рекомендуют интегрировать построение моделей с остальным ПО, использовать общие репозитории управления версиями для кодовых баз машинного обучения и отличных от них, и тесно связать их спринты.

### **4.3.1 Разнообразное восприятие**

Исследователи сгруппировали респондентов в три группы (низкая, средняя и высокая), равномерно распределенные по опыту работы в области ИИ. Во-первых, ранжировали каждую из отсортированных по карточкам категорий проблем респондентов, разделив их по категориям опыта с ИИ. Этот список представлен на рисунке 4.1, первоначально отсортированной по респондентам с низким опытом работы с ИИ.

Стоит обратить внимание на два момента. Во-первых, доступность данных, сбор, очистка и управление оцениваются респондентами как главная проблема, независимо от уровня опыта. Во-вторых, важность проблем определяется по мере восприятия респондентами искусственного интеллекта. Например, образование и профессиональная подготовка гораздо важнее для категории сотрудников с низким уровнем опыта работы с искусственным интеллектом. По мере приобретения сотрудниками (командами) опыта создания приложений, платформ и их интеграции в МО, повышаются навыки в восприятии и оценки проблем. Проблемы, связанные с инструментарием, масштабированием, эволюцией, оценкой и внедрением моделей, более важны для инженеров с большим опытом работы с ИИ, так как им в работе поручаются более сложные инженерные задачи.

Также сравнивается общая частота возникновения каждого вида проблем, используя те же три группы опыта работы в области искусственного интеллекта.

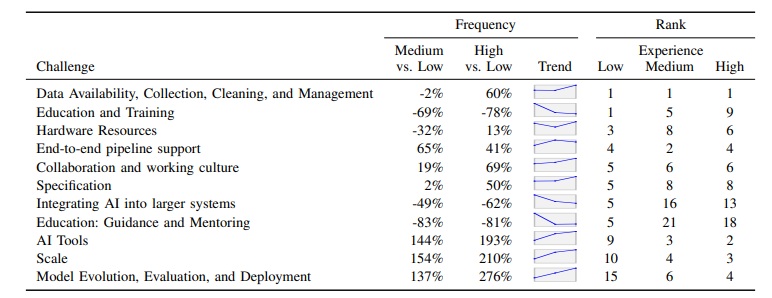


Рисунок 4.1 – Показатели рейтинга и личный опыт работы с ИИ

Если взглянуть на главную проблему рейтинга «Доступность, сбор, очистка и управление данными» то можно заметить, что в ней гораздо больше ответов (60%) дано респондентами с большим опытом работы. Аналогичная ситуация имеет место с проблемами, связанными со спецификациями. Однако, при рассмотрении вопросов «Образование и обучение», «Интеграция ИИ в более крупные системы» и «Образование: Руководство и наставничество», их частота значительно ниже по сравнению с респондентами с низким опытом работы. Интерпретируется это как менее важные проблемы для респондентов со средним и высоким опытом работы. Данная таблица показывает общую картину восприятия важности проблем в рамках определенной группы опыта и между группами.

Также проведен логистический регрессионный анализ для построения модели, которая могла бы объяснить различия в частоте при контроле личного опыта в области ИИ, опыта работы в команде, общего опыта работы, количества параллельных проектов в области ИИ, а также наличия у респондента образования в области МО. Выявлены пять значимых коэффициентов:

* образование и обучение отрицательно коррелировали с опытом работы в области ИИ с коэффициентом –0,18 (р <0,02), что означает, респонденты с меньшим опытом работы в области ИИ сочли этот вопрос важным;
* обучение других положительно коррелирует с личным опытом работы в области ИИ с коэффициентом 0,26 (р <0,01), что означает, респонденты с большим опытом работы в области ИИ сочли этот вопрос более важным;
* проблемы с инструментами положительно коррелируют с опытом работы команды в области ИИ с коэффициентом 0,13 (р <0,001), что означает, по мере накопления опыта работы команды в области ИИ, в которой они полагаются на взаимные инструменты;
* поддержка сквозного конвейера положительно коррелировала с формальным образованием (р <0,01), подразумевая создание конвейера респондентами с образованием;
* спецификации положительно коррелировали с образованием (р <0,03), подразумевая запись спецификации моделей и инженерных систем респондентами с образованием.

Опыт этих анализов заключается в том, что типы вопросов, которые инженеры считают важными, меняются по мере роста опыта работы в области ИИ. Отдельные проблемы носят временный характер, связаны с положением в команде и случайной сложностью совместной работы. Другие проблемы являются более фундаментальными для практики интеграции МО в программные приложения и затрагивают многих инженеров, независимо от уровня опыта работы [20].

### **4.3.2 На пути к модели зрелости процесса MО**

Ранее было отмечено о различиях в уровнях опыта работы с ИИ в командах разработчиков ПО. Это различие влияет на восприятие инженерных проблем, которые необходимо решать в повседневной практике. По мере становления и развития команд разработчиков ПО повышается эффективность и результативность предоставления основанных на МО рабочих продуктов и платформ. Для отражения зрелости MО, с учетом опыта работы была создана модель зрелости с шестью измерениями, оценивающими, каждую стадию рабочего процесса:

* имеет определенные цели;
* последовательно реализуется;
* документируется;
* автоматизируется;
* измеряется и отслеживается;
* постоянно совершенствуется.

Эти факторы основаны на концепции модели зрелости процесса, которая широко используется при разработке ПО и для оценки и повышения зрелости программных проектов.

В ходе опроса респонденты сообщили о зрелости двух этапов рабочего процесса, на которые каждый участник потратил больше времени (измеряется количеством часов, потраченным на каждую деятельность). Участники в ответах использовали формат ответов Лайкерта, от категорически не согласен (1) до категорически согласен (5):

* S1: у команды есть определенные цели, которых нужно достичь с помощью этой деятельности;
* S2: команда выполняет эту деятельность последовательно;
* S3: команда в значительной степени задокументировала практику, связанную с этой деятельностью;
* S4: команда выполняет эту работу в основном автоматизированным способом;
* S5: команда измеряет и отслеживает, насколько эффективно мы выполняем это задание;
* S6: команда постоянно совершенствует нашу практику, связанные с этой деятельностью.

Собраны данные по этапам специализировано знакомым респондентам на различных этапах рабочего процесса. Указанные данные учитывали работу респондентов с широким спектром применяемых методов MО. Вместо предписания (то есть cделать это, чтобы перейти на следующий уровень зрелости), проводилось описание (например, насколько автоматизирован и документирован этап рабочего процесса?).

Для анализа ответов был определен индекс зрелости деятельности, чтобы объединить отдельные оценки в единый показатель. Этот индекс представляет собой среднее значение согласия с шестью утверждениями зрелости S1…S6. В качестве средства проверки индекса зрелости проведена оценка эффективность, путем ответа на вопрос «Насколько эффективна, по вашему мнению, практика вашей команды в этой области по шкале от 1 (плохо) до 5 (отлично)?». Корреляция Спирмена между индексом зрелости и эффективности составила в диапазоне 0,4982 и 0,7627 (все статистически значимые при р <0,001) для всех видов деятельности искусственного интеллекта. Это подтверждает достоверность комплексного показателя индекс зрелости, который отражает зрелость и эффективность деятельности в области искусственного интеллекта.

В дополнение к индексу зрелости и эффективности деятельности получена оценка общей эффективности, на основе ответа респондентов на вопрос «Насколько эффективно ваша команда работает с ИИ по шкале от 1 (плохо) до 5 (отлично)». Наличие показателей индекса зрелости, эффективности деятельности и общей эффективности позволило сравнить зрелость и эффективность различных организаций, дисциплин и областей применения в компании Microsoft и определить направления развития. На рисунке 4.2 приведены указанные сравнения и показаны средние оценки общей эффективности, разделенные на девять наиболее актуальных областей применения ИИ.

Признано, что показатели представляют собой попытку количественной оценки показателя процесса, позволяющего командам оценить практику MО. В будущем необходимо совершенствование данного инструмента и дополнительной проверки его полезности.

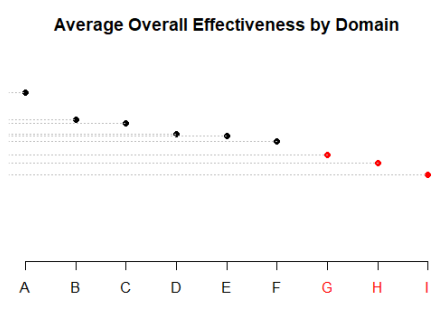


Рисунок 4.2 – Средняя общая эффективность методов MО

В данном разделе обобщены выводы о трех наблюдениях фундаментальных различий в разработке ПО, их адаптации для поддержки прошлых популярных прикладных областей и возможности адаптирования для поддержки приложений и платформ ИИ.

*Обнаружение и управление данными.*

Подобно тому как ПИ связана с программным кодом, который формирует ПО, также МО – это все, о данных на основе которых строятся модели обучения. Инженеры проектируют и создают системы, которые являются элегантными, абстрактными, модульными и простыми. В отличие от этого, данные, используемые в МО разнообразны, зависят от контекста, неоднородны и сложны в описании. Эти различия приводят к сложным проблемам, когда модели MО интегрируются в программные системы. Инженерам необходимо находить, собирать, контролировать, очищать и обрабатывать данные для использования в обучении и настройке моделей. Все данные должны храниться, отслеживаться и версионироваться. В то время как программные приложения описываются спецификациями, наборы данных редко имеют явные определения схем для описания столбцов и характеристики их статистических распределений. Однако из–за быстрой итерации, связанной с MО, схема данных (и данные) часто меняются, даже много раз в день. При необходимости изменения значений собираемых данных инженеры MО должны ждать обновления, развертывания и распространения инженерных систем, до поступления новых данных. Даже «простые» изменения могут оказать значительное влияние на объем собираемых данных, потенциально воздействуя на приложения через изменение характеристик производительности или увеличение использования пропускной способности сети. Наряду с существующими технологиями для создания версий кода, таковые отсутствуют для данных. Когда инженер собирает и обрабатывает данные, он может отслеживать эти неписаные детали, но при увеличении масштабов возможно возникновение проблемы поддержки знаний. Для кодифицирования информации в машиночитаемую форму необходимо использовать таблицы данных с применением надежного отслеживания характеристик метаданных этих наборов данных.

*Персонализация и повторное использование.*

Несмотря на известность требуемого времени для настройки и повторного использования компонентов кода, настройка моделей MО требует сравнительно большего времени. В ПО основными единицами повторного использования являются функции, алгоритмы, библиотеки и модули. Инженер находит исходный код библиотеки и вносит в него изменения, используя навыки, применяемые при разработке собственного ПО. Несмотря на техническую сложность работы обученные модели MО выглядят как функции, которые можно вызвать для заданных входных данных. Одной частью модели является алгоритм, на основе которого работает конкретный используемый метод МО, другой – это набор параметров, которые управляют функцией и изучаются в процессе обучения. При применении модели к предметной данной аналогичной области, на которых она была первоначально обучена, ее повторное использование несложно. Однако более существенные изменения необходимы, при запуске модели в другом домене или использование иного формата входных данных. Невозможно изменение параметры с помощью текстового редактора, так как требуется переобучение модели или ее полная замена. Оба варианта требуют от разработчиков ПО навыков МО. Кроме этого, переобучение или восстановление модели требует поиска, сбора и очистки дополнительных учебных данных, что потребует большого опыта и времени работы.

*Модульность MО.*

Следующим ключевым атрибутом проектирования крупномасштабных программных систем является модульность. Модули разделены и изолированы, для того чтобы разработка одного компонента не влияла на поведение других, находящихся в разработке. Также, модульность ПО усиливается законом Конвея, согласно которому команды, создающие каждый компонент ПО, организуются аналогично его архитектуре. Таким образом, отдельные модули часто поручаются отдельным командам. Взаимодействие модулей контролируется, выполняя двойную задачу, позволяя программным модулям оставаться отдельными, но также описывают интерфейсы, для минимизации количества коммуникаций, необходимых между отдельными командами, с целью обеспечения совместной работы. Поддержание строгих границ модулей между моделями МО затруднено по двум причинам. Во–первых, модели нелегко расширять. Например, нельзя взять модель NLP для английского языка и добавить отдельную модель NLP для заказа пиццы и ожидать их правильной совместной работы. Модели должны вместе разрабатываться и обучаться. Во–вторых, модели взаимодействуют неочевидными способами. В крупномасштабных системах с более чем одной моделью результаты каждой модели будут влиять на процессы обучения и настройки друг друга. Фактически, эффективность одной модели изменится в результате использования другой модели, даже если их код будет разделен. Таким образом, даже если бы каждую модель создавали отдельные команды, им пришлось бы тесно сотрудничать, чтобы должным образом обучать или поддерживать работоспособность системы. Это явление (также называемое запутанностью компонентов) может привести к немонотонному распространению ошибок, что означает, улучшения в одной части системы могут снизить общее качество системы, поскольку остальная часть системы не настроена на последние улучшения. Эта проблема становится еще более очевидной в тех случаях, когда модели МО обновляются несовместимым образом и вносят новые, ранее невиданные ошибки, которые нарушают взаимодействие с другими частями системы.

### **4.3.3 Ограничения**

Тематическое исследование проведено с участием команд компании Microsoft, крупной мировой компании-разработчика программного обеспечения с разнообразным портфелем программных продуктов. Она также является одним из крупнейших поставщиков продуктов и платформ на МО. Некоторые результаты специфичны для команд Microsoft и членов других команд, участвовавших в интервью и опросах. Тем не менее, учитывая большое разнообразие проектов, представленных информантами возможно применение данного опыта другими компаниями. Отдельные выводы зависят от конкретного рабочего процесса MО, используемого некоторыми командами разработчиков ПО в Microsoft. Результаты интервью и опросов основаны на самостоятельно отобранных информантах и данных, предоставленных самими респондентами [20].

# ЗАКЛЮЧЕНИЕ

Машинное обучение применяется к широкому кругу экономических задач – от обнаружения мошенничества до выбора целевой аудитории и рекомендаций товара, наблюдения за производством в реальном вре­мени, анализа тональности текстов и медицинской диагностики. Оно может взять на себя задачи, которые невозможно выполнить вручную из–за огромного количества подлежащих обработке данных.

Машинное обучение на фоне традиционных подходов показывает свою практичность в применении, обеспечивает точность, автоматизированность, оперативность и объемность рассматриваемого информационного продукта, чем подтверждает актуальность своего использования в настоящем времени и гибкость дальнейшего развития в будущем прогрессивном обществе.

Программная инженерия, являясь продуктом ИТ отрасли представляет комплекс систем планирования, разработки, эксплуатации и сопровождения ПО. Она охватывает все этапы создания ПО от начала формулирования требований, разработки, сопровождения и окончательного его списания. Теоретическая основа и практическое применение ПИ базируется на современных теориях программирования и менеджмента, имеет огромный потенциал дальнейшего развития в направлении совершенствования информационных и других технологий.

Многие команды компании Microsoft приложили значительные усилия для разработки обширного портфеля приложений и платформ для ИИ, интегрируя МО в существующие процессы разработки ПО, и создания условий для развития талантливых специалистов в области ИИ.

В качестве ключевого вывода анализа были обнаружены три фундаментальных отличия в создании приложений и платформ для обучения и внедрения моделей МО, которые видели в предыдущих областях применения. Во-первых, МО – это все о данных. Количество усилий и строгость, которые требуются для обнаружения, получения, управления и создания версий, по своей сути сложны. Во-вторых, создание настраиваемых и расширяемых моделей требует от команд не только навыков разработки ПО, но и требуется достаточно глубокое знания в области МО, для построения, оценки и настройки моделей с нуля. В-третьих, поддержание строгих границ модулей между компонентами МО, гораздо сложнее модулей ПИ. Модели МО могут быть сложными, что приводит к их взаимному влиянию в процессе обучения и настройки.

Основываясь на результатах исследования группы ученых, необходимо отметить, что МО является неотъемлемой частью ИИ в виде отдельного его развития. На основе указанных результатов обобщен набор практик для решения проблем, имеющих фундаментальное значение для крупномасштабной разработки и внедрения приложений на основе MО. Одни проблемы были связаны с опытом респондентов в области ИИ, в то время как другие были применимы к большинству респондентов, создающих приложения ИИ. В заключении определены три аспекта в области ИИ, которые делают ее принципиально отличной от предыдущих областей применения. Их влияние потребует значительных исследовательских усилий для рассмотрения в будущем.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Машинное обучение / Хенрик Бринк, Джозеф Ричардс, Марк Феверолф, Санкт-Петербург, 2017. – 338 с.
2. Машинное обучение / Том Митчелл, Москва, 1997. – 220 с.
3. For Big–Data Scientists, Janitor Work is Key Hurdle to Insights, / Steve Lohr, New York, 2014. – 26 с.
4. One Jump Ahead: Computer Perfection at Checkers / Jonathan Schaeffer, New York, 2009. – 192 с.
5. Собаки против кошек / [Электронные ресурс] URL: www.kaggle.com/c/dogs–vs–cats (Дата обращения 15.05.2022).
6. German Credit Data / [Электронные ресурс] URL: http://mng.bz/95r4 (Дата обращения 15.05.2022).
7. G. James «An Introduction to Statistical Learning» / [Электронные ресурс] URL: http://www-bcf.usc.edul-gareth (Дата обращения 16.05.2022).
8. Программная инженерия. Базовые основы программной инженерии / Е.М. Лаврищева, МФТИ, 2016. – 51 с.
9. Программная инженерия. Основные понятия и определения / Е.М. Лаврищева, Киев, 1989. – 36 с.
10. Инженерное проектирование программного обеспечения / Б.У. Боем, Москва, 1986. – 510 с.
11. Software engineering as a scientific and engineering discipline / E.M. Lavrishcheva, Moscow, 2008. – P. 332/
12. Фабрики разработки программ / Дж. Гринфильд, СПб–Киев, 2007. – 591 с.
13. Инженерия программного обеспечения / Н. Соммервилл, СПб. –Киев, 2002. – 623 с.
14. Машинное обучение в программной инженерии / [Электронные ресурс] https://www.microsoft.com/en–us/research/uploads/prod/2019/03/amershi–icse–2019 (Дата обращения 16.05.2022).
15. Microsoft AI Platform, in Deep Learning with Azure / M. Salvaris, D. Dean, and W. H. Tok, New York, 2018. – p. 98.
16. Usage and perceptions of agile software development in an industrial context: An exploratory study / A. Begel, N. Nagappan, New York, 2007. – p. 25.
17. Pair programming: What’s in it for me? / A. Begel, N. Nagappan, New York, 2008. – p. 128.
18. Have agile techniques been the silver bullet for software development at microsoft? / B. Murphy, C. Bird, T. Zimmermann, L. Williams, N. Nagappan, A. Begel, New York, 2013. – p. 84.
19. DevOps capabilities, practices, and challenges: Insights from a case study / M. Senapathi, J. Buchan, H. Osman, New York, 2018. – p. 67.
20. Maintaining mental models: A study of developer work habits / T. D. LaToza, G. Venolia, R. DeLine, New York, 2006. – p. 501.