

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

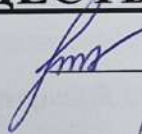
Магистерская программа: «Прикладной искусственный интеллект»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

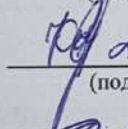
на тему

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА
АВТОМАТИЗАЦИИ РАБОТЫ ЗАВЕДЕНИЙ
ОБЩЕСТВЕННОГО ПИТАНИЯ

Студент

 25.05.2026 Мартынов Никита Владимирович
(подпись, дата) (ФИО полностью)

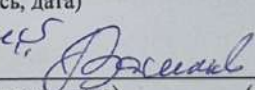
Руководитель

 25.05.2026 Карамышева Н.С.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

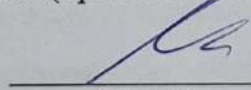
 25.05.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

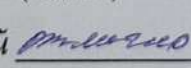
Рецензент

заведующий кафедрой
АО «ЧПП» РЭО-ИИ  25.05.2026 Васильев А.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 № 14)

Заведующий кафедрой

 М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от 09.06.2026 № 17)

Секретарь ГЭК

 Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«УТВЕРЖДАЮ»
Заведующий кафедрой
М.А. Митрохин
« 20 » 09 2024 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Мартынов Никита Владимирович гр. 24ВВИм1 факультета ВТ
2. Тема работы «Рекомендательная система автоматизации работы заведений общественного питания»

Тема утверждена приказом ПГУ №920/о от "20" сентября 2024г.

3. Руководитель магистерской диссертации к.т.н., доцент Карамышева Н.С.

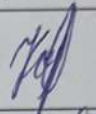
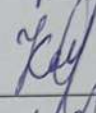
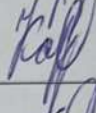
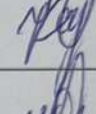
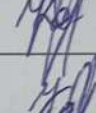
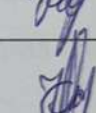
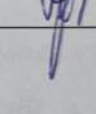
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: разработка интеллектуальной рекомендательной системы заведений общественного питания.

Разрабатываемая система должна выполнять автоматическое формирование рекомендаций объектов меню на основе анализа пользовательских предпочтений, пользовательских заказов и характеристик блюд.

5. Перечень подлежащих разработке вопросов

1. Анализ предметной области.
2. Изучение литературы по теме исследования.
3. Разработка модуля генерации персональных предложений.
4. Тестирование разработанной системы, анализ полученных результатов.
5. Подтверждение гипотезы исследования.
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	11%	20.09.2024- 29.01.2025	
2.	Анализ требований	11%	29.01.2025- 30.03.2025	
3.	Изучение литературы по теме исследования	22%	30.03.2025- 30.05.2025	
4.	Проектирование и реализация системы	27%	01.06.2025- 01.09.2025	
5.	Тестирование	17%	02.09.2025- 02.10.2025	
6.	Анализ полученных результатов исследования	6%	01.04.2026- 29.04.2026	
7.	Оформление пояснительной записки	6%	30.04.2026- 25.05.2026	

Дата выдачи задания " 20 " 09 2024 г.

Руководитель магистерской диссертации
Н.С.

(подпись, дата)

Задание к исполнению принял студент
Н.В.

(подпись, дата)

Работу к защите допустить
Декан факультета

(подпись, дата)

Карамышева

Мартынов

Фионова Л.Р.

РЕФЕРАТ

Магистерская диссертация содержит 122 листов, 28 рисунков, 19 таблиц, 13 источников, 2 приложения.

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, МАШИННОЕ ОБУЧЕНИЕ, ПЕРСОНАЛИЗАЦИЯ, SVD, kNN, CONTENT-BASED FILTERING, MACHINE LEARNING, POS-SYSTEM

Основной целью работы является исследование и разработка рекомендательной системы для предприятий общественного питания малого масштаба, обеспечивающей формирование персональных рекомендаций в условиях ограниченного объема данных и вычислительных ресурсов.

Объектом исследования являются процессы формирования рекомендаций в системах автоматизации предприятий общественного питания.

Предметом исследования являются методы и алгоритмы построения рекомендательных систем для предприятий общественного питания малого масштаба.

В рамках работы выполнен анализ существующих рекомендательных систем, рассмотрены особенности применения рекомендательных технологий в сфере общественного питания, а также исследованы методы коллаборативной фильтрации, content-based рекомендаций и метод k-ближайших соседей.

В работе выполнена подготовка данных для формирования рекомендаций, проведено экспериментальное сравнение выбранных моделей и определены области их эффективного применения.

Результаты исследования могут быть использованы при разработке систем автоматизации предприятий общественного питания, ориентированных на персонализацию обслуживания клиентов и повышение эффективности продаж.

ПГУ1.090401.15.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата	Рекомендательная система автоматизации работы заведений общественного питания	Лит.	Лист	Листов
Разраб.		Мартынов Н.В.		15.05.16				
Провер.		Карамышева Н.С.		15.05.16			2	122
Н. Контр		Бычков А.С.		15.05.16		ФВТ, каф. ВТ гр. 24ВВИм1		
Утверд.		Митрохин М.А.		15.05.16				

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения и обозначения:

БД – База данных

ИИ – Искусственный интеллект

МО – Машинное обучение

ПО – Программное обеспечение

РС – Рекомендательная система

AI (Artificial Intelligence) – Искусственный интеллект

API (Application Programming Interface) – Программный интерфейс приложения

CB (Content-Based Filtering) – Контентно-ориентованная фильтрация

CRM (Customer Relationship Management) – Система управления взаимоотношениями с клиентами

EDA (Exploratory Data Analysis) – Разведочный анализ данных

JSON (JavaScript Object Notation) – Текстовый формат обмена данными

kNN (k-Nearest Neighbors) – Метод k-ближайших соседей

ML (Machine Learning) – Машинное обучение

POS (Point of Sale) – Система автоматизации точки продаж

SQL (Structured Query Language) – Язык структурированных запросов

SVD (Singular Value Decomposition) – Сингулярное разложение матрицы

ВВЕДЕНИЕ

В условиях активного развития цифровых технологий и автоматизации процессов всё большее распространение получают рекомендательные системы, применяемые в различных сферах деятельности человека [1, 4]. Подобные системы используются в электронной коммерции, мультимедийных сервисах, социальных сетях и сфере услуг для персонализации предложений и повышения удобства взаимодействия пользователей с информационными системами.

Особую актуальность рекомендательные системы приобретают в сфере общественного питания, где они позволяют автоматизировать процесс подбора блюд, повысить качество обслуживания клиентов и увеличить эффективность работы заведений [5]. Использование рекомендательных механизмов способствует формированию персональных предложений, сокращению времени выбора блюд и увеличению среднего чека.

В настоящее время большинство существующих рекомендательных решений ориентировано преимущественно на крупные онлайн-платформы и сервисы доставки еды. Однако предприятия общественного питания малого масштаба обладают рядом особенностей, включая ограниченный объём пользовательских данных, ограниченные вычислительные ресурсы и необходимость обеспечения простоты использования системы. Это затрудняет применение сложных рекомендательных платформ в условиях малого бизнеса и обуславливает необходимость разработки адаптированных решений.

Темой магистерской диссертации является исследование методов построения рекомендательных систем для автоматизации работы предприятий общественного питания малого масштаба.

Основной целью работы является разработка и исследование рекомендательной системы, обеспечивающей формирование персональных и контекстных рекомендаций в условиях ограниченного объёма данных и вычислительных ресурсов.

Гипотезой исследования является предположение о том, что использование гибридного подхода, основанного на сочетании методов коллаборативной фильтрации, content-based рекомендаций и метода k-ближайших соседей, позволит повысить качество формирования рекомендаций для различных сценариев взаимодействия с пользователем в условиях малого общепита [13].

Объектом исследования в магистерской диссертации являются процессы формирования рекомендаций в системах автоматизации предприятий общественного питания.

Предметом исследования являются методы и алгоритмы построения рекомендательных систем для предприятий общественного питания малого масштаба.

Основными задачами работы являются:

- анализ существующих подходов к построению рекомендательных систем,
- исследование особенностей применения рекомендательных систем в условиях малого общепита,
- анализ существующих решений и выявление их ограничений;
- выбор методов формирования рекомендаций для различных сценариев взаимодействия с пользователем,
- подготовка данных для обучения и тестирования моделей,
- реализация и исследование моделей рекомендаций на основе методов SVD, content-based и k-ближайших соседей,
- проведение экспериментального сравнения выбранных моделей,
- анализ полученных результатов и определение областей эффективного применения рассматриваемых подходов.

Магистерская диссертация направлена на исследование и разработку рекомендательной системы, адаптированной к условиям функционирования предприятий общественного питания малого масштаба.

1 Анализ предметной области и постановка задачи

1.1 Назначение рекомендательных систем

В условиях роста объёма доступной информации пользователи сталкиваются с необходимостью выбора наилучшего предложения среди большого количества товаров и услуг. Это усложняет процесс принятия решений и увеличивает время, затрачиваемое на поиск подходящих вариантов. В связи с этим возрастает роль рекомендательных систем, позволяющих автоматизировать процесс подбора релевантных предложений.

Рекомендательная система – это программный инструмент, предназначенный для формирования персональных предложений пользователю на основе анализа его предпочтений, поведения или характеристик объектов [1]. Основной целью использования таких систем является предоставление пользователю наиболее подходящих товаров или услуг с учётом его индивидуальных особенностей с целью повышения лояльности посетителя [4].

Рекомендательные системы находят применение в различных сферах, таких как электронная коммерция, мультимедийные сервисы, социальные сети и сфера услуг. В частности, в предприятиях общественного питания они могут использоваться для предложения блюд, соответствующих предпочтениям клиента, а также для формирования дополнительных рекомендаций.

К основным задачам рекомендательных систем относятся:

- персонализация предложений,
- помощь пользователю в выборе,
- сокращение времени поиска подходящих вариантов,
- увеличение среднего чека,
- ускорение процесса обслуживания клиентов [1, 12].

Таким образом, рекомендательные системы позволяют повысить удобство взаимодействия пользователя с системой и обеспечить более эффективное предоставление услуг за счёт формирования релевантных предложений.

1.2 Особенности применения рекомендательных систем в сфере общественного питания

Предприятия общественного питания малого масштаба существенно отличаются от крупных сетевых заведений и онлайн-платформ как по объёму доступных данных, так и по используемой инфраструктуре. Эти особенности необходимо учитывать при разработке рекомендательных систем, так как классические подходы, применяемые в крупных сервисах, не всегда эффективны в условиях малого бизнеса [5].

Одной из ключевых особенностей является ограниченный объём данных. В небольших заведениях число клиентов и заказов относительно невелико, что приводит к недостаточности информации для построения сложных моделей рекомендаций. Кроме того, в большинстве случаев отсутствуют явные оценки пользователей, такие как рейтинги или отзывы, что затрудняет использование традиционных методов анализа предпочтений.

Важным фактором также являются ограничения вычислительных ресурсов. В малых заведениях чаще всего используются простые кассовые или POS-терминалы, не обладающие высокой производительностью [7]. Это накладывает ограничения на сложность применяемых алгоритмов и требует использования лёгких и быстрых методов обработки данных.

Дополнительным ограничением выступает ограниченный бюджет на внедрение и сопровождение информационных систем. Малые предприятия не всегда могут позволить себе использование дорогостоящих решений, требующих развертывания серверной инфраструктуры или привлечения специалистов для настройки и поддержки.

Не менее важным является фактор удобства использования системы. Персонал заведения, как правило, не обладает специализированными техническими знаниями, поэтому интерфейс и логика работы системы должны быть максимально простыми и понятными, а обучение сотрудников – минимальным.

Также следует учитывать требования к скорости работы системы. В процессе обслуживания клиента рекомендации должны формироваться практически мгновенно, без задержек, чтобы не замедлять процесс оформления заказа.

Отдельного внимания заслуживает зависимость от интернет-соединения. В ряде случаев стабильный доступ к сети может отсутствовать или быть ограниченным по причине расположения подобных заведений вне крупных населенных пунктов, где у провайдеров предоставляемые услуги фиксированного (проводного) или беспроводного доступа к сети зачастую не соответствуют современным требованиям по скорости приема/передачи данных. По этой причине приведенные ограничения делают актуальной возможность работы системы в офлайн-режиме или с минимальной зависимостью от внешних сервисов.

Таким образом, особенности функционирования предприятий общественного питания малого масштаба обуславливают необходимость разработки рекомендательных систем, которые обладают следующими характеристиками: простота реализации и использования, низкие требования к вычислительным ресурсам, способность работать с ограниченным объёмом данных и обеспечивать быстрый отклик [5].

1.3 Анализ существующих решений

Приложение доставки еды из магазинов и ресторанов «Яндекс Еда»

Одним из наиболее распространённых сервисов в сфере доставки еды является платформа «Яндекс Еда». Данный сервис предоставляет пользователям возможность заказа блюд из различных ресторанов и использует встроенные рекомендательные механизмы для повышения удобства выбора и увеличения количества заказов.

Рекомендательные функции в системе реализованы на различных этапах взаимодействия пользователя с приложением: при поиске, выборе блюд и оформлении заказа. Основной задачей таких механизмов является

персонализация предложений и формирование релевантных рекомендаций с учётом предпочтений пользователя.

На главной странице приложения пользователю отображаются блоки с рекомендациями, сформированными с учётом его предпочтений и текущих условий. К таким блокам относятся «Предложения для вас», тематические категории блюд (например, бургеры, пицца, суши), а также акции и персональные скидки (рисунок 1).



Рисунок 1 – Персональные предложения на главной странице «Яндекс Еда»

В «Яндекс Еде» реализован интеллектуальный помощник, позволяющий формировать рекомендации на основе текстовых запросов пользователя. Доступен выбор желаемых характеристик блюда или условий заказа, после чего система предлагает релевантные варианты (рисунок 2).

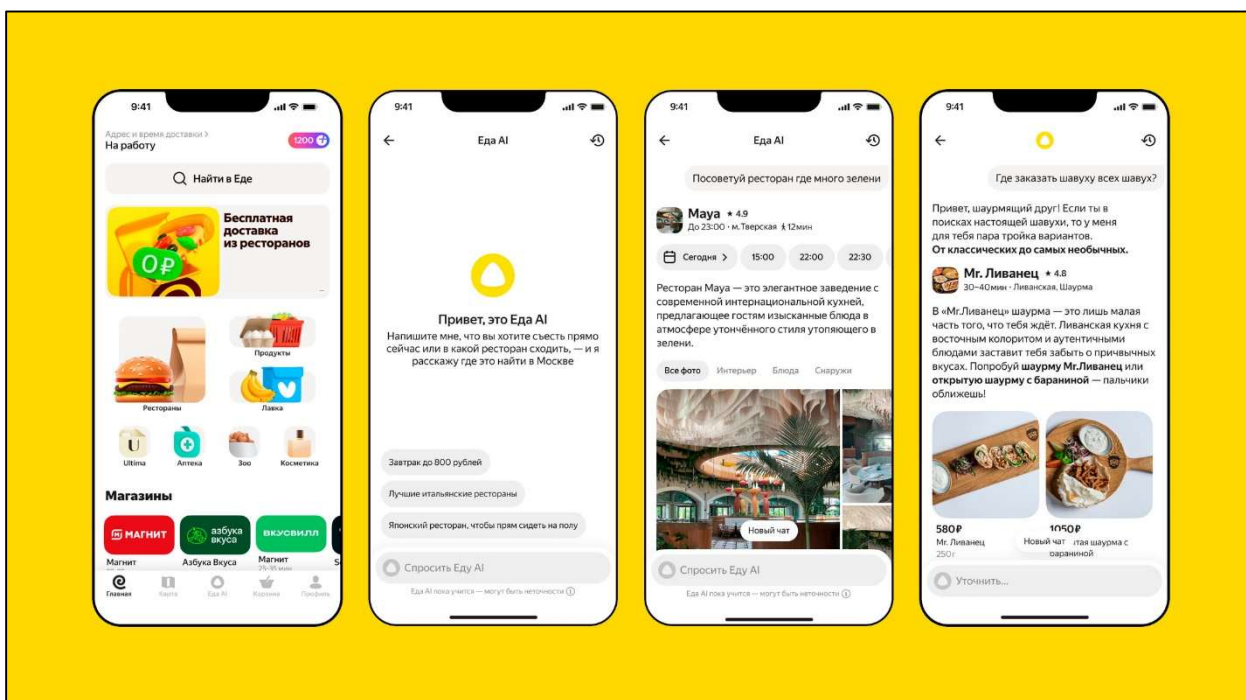


Рисунок 2 – Интеллектуальный помощник «Еда AI»

На этапе формирования заказа пользователю предлагаются дополнительные позиции в блоке «Что-то ещё?», а также в аналогичных разделах (рисунок 3). Такие рекомендации формируются на основе анализа поведения других пользователей и позволяют предлагать товары, которые часто заказываются вместе [2, 9].

При оформлении заказа система дополнительно предлагает товары, которые могут дополнить текущий выбор пользователя. Это позволяет реализовать механизм дополнительных продаж и увеличить итоговую стоимость заказа.

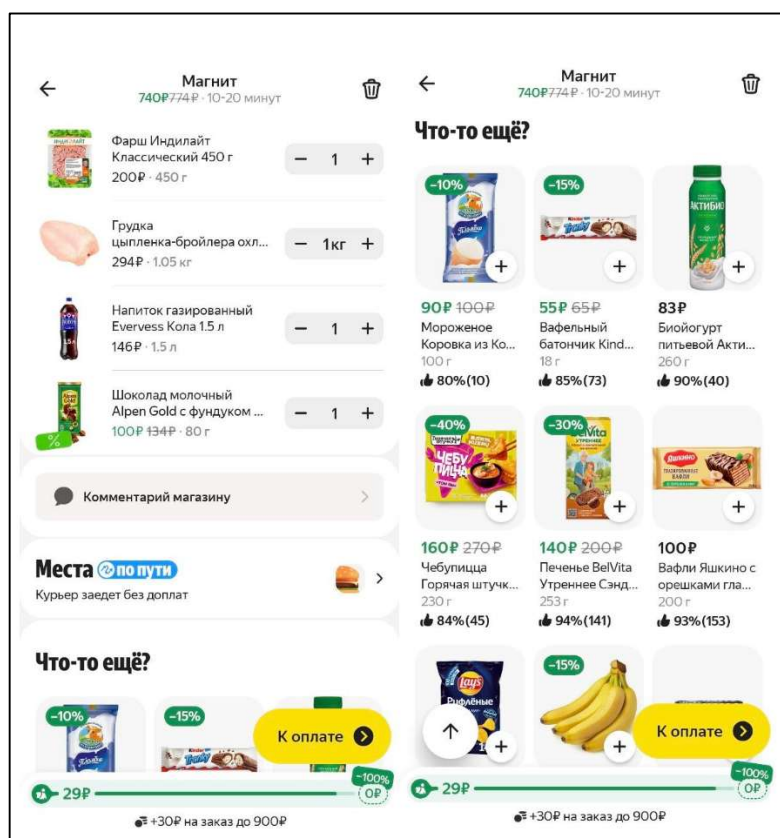


Рисунок 3 – Рекомендации сопутствующих товаров в приложении «Яндекс Еда»

К преимуществам платформы «Яндекс Еда» можно отнести:

- высокий уровень персонализации рекомендаций,
- интеграция рекомендаций на всех этапах взаимодействия,
- использование современных технологий ИИ,
- удобный и интуитивно понятный интерфейс,
- высокая эффективность при большом объеме данных.

Среди недостатков приложения доставки еды можно выделить:

- высокая сложность реализации системы,
- зависимость от большого объема пользовательских данных,
- необходимость постоянного интернет-соединения,
- значительные требования к вычислительным ресурсам,
- сложность адаптации к условиям малого бизнеса.

Несмотря на высокий уровень развития и эффективности рекомендательных механизмов, реализованных в системе «Яндекс Еда», подобные решения ориентированы на крупные цифровые платформы и

требуют значительных ресурсов. Это ограничивает возможность их использования в условиях предприятий общественного питания малого масштаба.

Система автоматизации заказов с элементами рекомендаций «r_keeper Kiosk Pro»

Система r_keeper Kiosk Pro представляет собой решение для автоматизации процесса оформления заказов в предприятиях общественного питания, реализованное в виде терминалов самообслуживания [6]. Данная система позволяет пользователям самостоятельно выбирать блюда, формировать заказ и осуществлять оплату без участия кассира.

Основной задачей системы является ускорение обслуживания клиентов, снижение нагрузки на персонал, а также увеличение среднего чека за счёт внедрения механизмов дополнительных продаж.

Рекомендательные механизмы в системе r_keeper Kiosk Pro реализованы на основе заранее заданных правил и сценариев, которые настраиваются в административной части системы [7].

В системе предусмотрен механизм отображения рекомендованных блюд, который активируется при выборе пользователем определённой позиции меню. В этом случае пользователю предлагаются дополнительные блюда, связанные с выбранным товаром (рисунок 4) [7].

Как показано на рисунке 4, рекомендации отображаются в виде всплывающего окна с набором заранее заданных позиций. Формирование таких рекомендаций осуществляется вручную: для каждого блюда задаётся список связанных товаров, которые будут предложены пользователю.

Более сложным механизмом являются цепочки рекомендаций, представляющие собой последовательность шагов, на каждом из которых пользователю предлагаются дополнительные товары или категории (рисунок 5) [8].

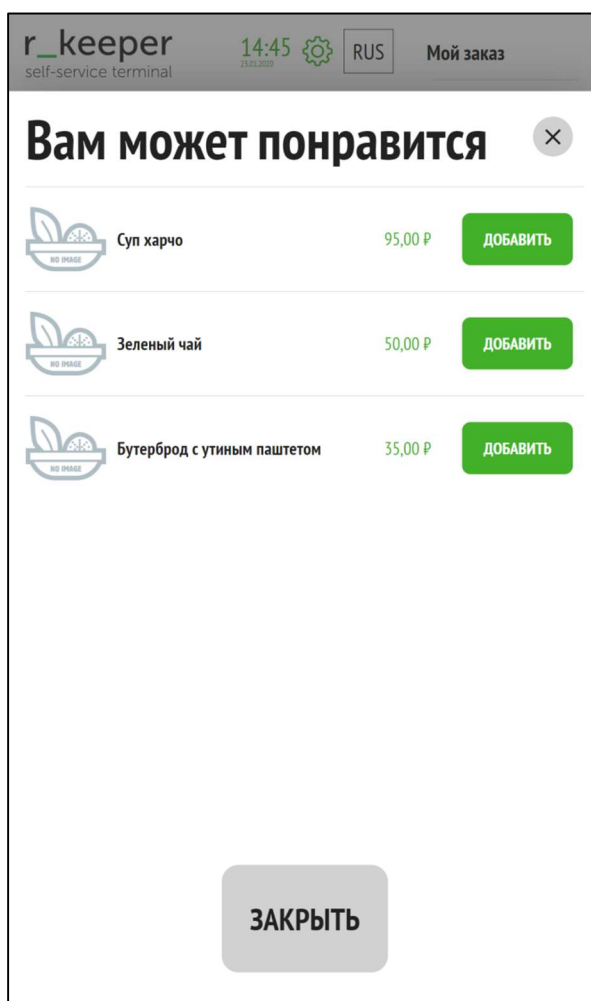


Рисунок 4 – Рекомендации блюд на этапе оформления заказа

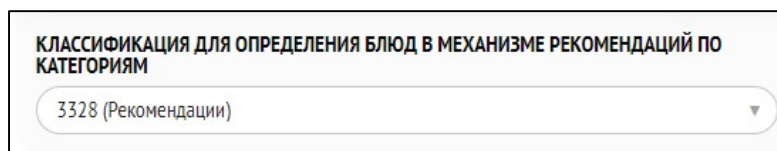


Рисунок 5 – Настройка цепочки рекомендаций в «r_keeper Kiosk Pro»

Цепочки рекомендаций позволяют формировать сценарии последовательного предложения товаров. Например, если пользователь добавил в заказ основное блюдо, тогда к нему можно рекомендовать различные соусы и напитки из меню заведения.

Каждый этап цепочки настраивается вручную и может включать как отдельные блюда, так и категории. Запуск цепочки может быть привязан к выбору определённой позиции меню.

Механизмы рекомендаций в системе реализуются через сценарии, определяющие условия запуска рекомендаций, состав предлагаемых блюд и порядок их отображения (рисунок 6). Система предоставляет возможность

гибкой настройке логики рекомендаций, однако требует предварительного конфигурирования со стороны администратора [8].

Рисунок 6 – Настройка сценария рекомендация в r_keeper Kiosk Pro

К преимуществам данной системы можно отнести:

- автоматизация процесса оформления заказов,
- сокращение времени обслуживания клиентов,
- увеличение среднего чека за счёт дополнительных предложений,
- возможность настройки сценариев рекомендаций,
- интеграция с системой управления рестораном.

Среди недостатков системы можно выделить:

- рекомендации формируются вручную и не адаптируются автоматически,
- отсутствует персонализация на уровне пользователя,
- нет использования методов машинного обучения,
- зависимость от предварительной настройки сценариев,
- необходимость привлечения специалистов для настройки,
- высокая стоимость внедрения и сопровождения,

– избыточность функционала для малого бизнеса.

Система r_keeper Kiosk Pro реализует рекомендации на основе заранее заданных сценариев и правил, что позволяет эффективно решать задачи увеличения среднего чека. Однако отсутствие автоматической персонализации и зависимость от ручной настройки ограничивают её применение как полноценной рекомендательной системы, особенно в условиях предприятий общественного питания малого масштаба.

Рекомендательная система Amazon

Одним из наиболее известных примеров применения рекомендательных систем является платформа Amazon. Рекомендательные механизмы играют ключевую роль в работе сервиса и используются для персонализации пользовательского опыта, повышения удобства поиска товаров и увеличения объёма продаж [1, 12].

Система рекомендаций Amazon анализирует действия пользователей, включая историю просмотров, покупок, поисковые запросы и взаимодействия с товарами. На основе этих данных формируются персональные предложения, отображающиеся на различных этапах взаимодействия пользователя с платформой.

В системе Amazon рекомендации интегрированы непосредственно в интерфейс платформы и представлены в виде различных блоков.

Одним из наиболее известных механизмов является блок рекомендаций «Frequently Bought Together» («С этим товаром часто покупают»), отображаемый в карточке товара (рисунок 7) [12].

Как показано на рисунке 7, система предлагает пользователю товары, которые часто приобретаются совместно с выбранным товаром другими пользователями. Подобный подход реализует механизм cross-sell и позволяет увеличивать итоговую стоимость заказа [1].

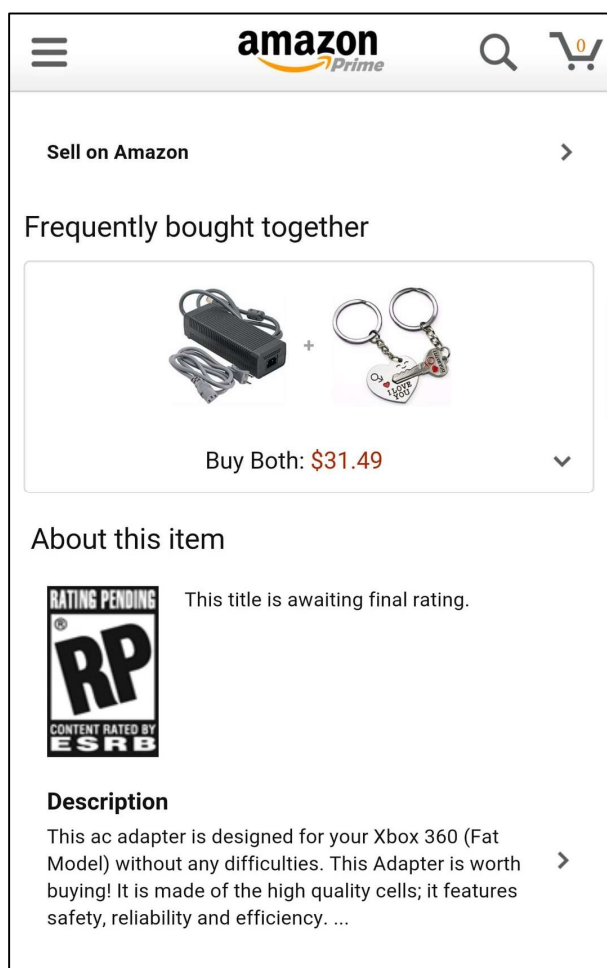


Рисунок 7 – Блок рекомендаций «Frequently Bought Together» («С этим товаром часто покупают») в сервисе Amazon

Дополнительно Amazon использует блоки персональных предложений, формируемых на основе поведения конкретного пользователя.

К таким блокам относятся «Recommended for You» («Рекомендовано для вас»), «Customers who bought this item also bought» («С этим товаром также покупают»), «Inspired by your browsing history» («На основе ваших просмотров») [12].

К преимуществам рекомендательной система Amazon можно отнести:

- высокий уровень персонализации рекомендаций,
- автоматическое обновление рекомендаций,
- высокая точность при большом объеме данных,
- интеграция рекомендаций на различных этапах взаимодействия пользователя с системой,
- масштабируемость решений.

Среди недостатков можно выделить:

- необходимость наличия большого объёма пользовательских данных;
- высокие требования к вычислительным ресурсам,
- сложность реализации и сопровождения,
- зависимость от развитой серверной инфраструктуры,
- ограниченная применимость в условиях малого бизнеса,
- отсутствие адаптации под особенности предприятий общественного питания.

Рекомендательная система Amazon демонстрирует высокую эффективность за счёт использования методов анализа поведения пользователей и коллаборативной фильтрации. Однако подобные решения ориентированы на крупные платформы и требуют значительных вычислительных ресурсов и объёма данных, что ограничивает возможность их применения в условиях предприятий общественного питания малого масштаба.

Несмотря на различия в используемых подходах, рассмотренные системы обладают рядом общих ограничений. Одной из основных проблем современных рекомендательных систем является зависимость качества рекомендаций от объёма накопленных пользовательских данных. При недостаточном количестве информации эффективность рекомендаций существенно снижается [1, 12].

Дополнительной проблемой является так называемая проблема холодного старта, возникающая при отсутствии данных о новых пользователях или новых товарах. В подобных условиях многие методы рекомендаций не способны формировать качественные персональные предложения [1, 2].

Также существенным ограничением является высокая сложность современных рекомендательных решений. Многие системы требуют значительных вычислительных ресурсов, развитой серверной инфраструктуры и постоянного интернет-соединения, что затрудняет их использование в условиях предприятий общественного питания малого масштаба.

Кроме того, существующие решения зачастую ориентированы либо на крупные онлайн-платформы, либо на автоматизацию внутренних бизнес-процессов, не обеспечивая при этом достаточного уровня гибкости, персонализации и простоты внедрения для небольших заведений.

В связи с этим возникает необходимость разработки рекомендательной системы, адаптированной к условиям функционирования предприятий общественного питания малого масштаба и учитывающей ограничения, связанные с объёмом данных, вычислительными ресурсами и особенностями процесса обслуживания клиентов.

1.4 Постановка задачи исследования

Проведённый анализ существующих рекомендательных систем показал, что современные подходы к формированию рекомендаций обладают рядом ограничений при применении в условиях предприятий общественного питания малого масштаба. К основным проблемам относятся зависимость от большого объёма пользовательских данных, высокая сложность реализации, необходимость использования вычислительных ресурсов и ограниченная адаптация к условиям малого бизнеса.

Кроме того, существующие решения либо ориентированы на крупные онлайн-платформы с развитой инфраструктурой, либо используют заранее заданные сценарии рекомендаций, не обеспечивающие достаточного уровня персонализации.

В связи с этим возникает необходимость разработки рекомендательной системы, адаптированной к особенностям функционирования предприятий общественного питания малого масштаба и способной обеспечивать формирование рекомендаций в условиях ограниченного объёма данных и вычислительных ресурсов.

Объектом исследования являются процессы формирования рекомендаций в системах автоматизации предприятий общественного питания.

Предметом исследования являются методы и алгоритмы построения рекомендательных систем для предприятий общественного питания малого масштаба.

Целью данной работы является разработка и исследование рекомендательной системы для предприятий общественного питания малого масштаба, обеспечивающей формирование персональных и контекстных рекомендаций на различных этапах взаимодействия с клиентом.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить анализ существующих подходов к построению рекомендательных систем,
- исследовать особенности применения рекомендательных систем в условиях предприятий общественного питания малого масштаба,
- выполнить сравнительный анализ существующих решений и выявить их ограничения,
- выбрать методы формирования рекомендаций для различных сценариев взаимодействия с пользователем,
- подготовить данные для обучения и тестирования моделей,
- реализовать и исследовать модели рекомендаций на основе методов SVD, content-based и k-ближайших соседей [1, 2, 3],
- провести экспериментальное сравнение выбранных моделей,
- выполнить анализ полученных результатов и определить области эффективного применения рассматриваемых подходов.

2 Обзор методов построения рекомендательных систем и выбор моделей

2.1 Коллаборативная фильтрация и метод SVD

Рекомендательные системы, основанные на коллаборативной фильтрации, являются одним из наиболее распространённых подходов к формированию персональных рекомендаций [1]. Основная идея данного подхода заключается в анализе поведения пользователей и выявлении сходства между их предпочтениями. Предполагается, что пользователи с похожей историей взаимодействий могут быть заинтересованы в схожих объектах.

Коллаборативная фильтрация не требует анализа содержимого объектов и основывается исключительно на данных о взаимодействии пользователей с системой, например, на истории заказов, просмотров или оценок [4]. Это позволяет выявлять скрытые закономерности в поведении пользователей и формировать персональные рекомендации.

Существует несколько основных подходов к реализации коллаборативной фильтрации [2]:

- user-based подход,
- item-based подход,
- методы матричной факторизации.

В user-based подходе рекомендации формируются на основе поиска пользователей со схожими предпочтениями. Если два пользователя заказывали похожие блюда, то система может рекомендовать одному пользователю блюда, выбранные другим.

Item-based подход основывается на поиске схожих объектов. В данном случае рекомендации формируются на основе анализа того, какие блюда чаще всего выбираются совместно или имеют схожий характер взаимодействия со стороны пользователей.

Одним из наиболее распространённых современных подходов к реализации коллаборативной фильтрации являются методы матричной факторизации [9]. Их основная идея заключается в представлении данных о

взаимодействии пользователей и объектов в виде матрицы и последующем выделении скрытых факторов, влияющих на предпочтения пользователей.

В рамках методов матричной факторизации информация о взаимодействии пользователей с объектами представляется в виде матрицы «пользователь-объект», где строки соответствуют пользователям, а столбцы – объектам рекомендаций [2]. В контексте данной работы в качестве объектов выступают блюда предприятия общественного питания, а значения матрицы отражают информацию о взаимодействии пользователя с конкретным блюдом (таблица 1).

Таблица 1 – Пример матрицы взаимодействия пользователей и объектов

Пользователь	Блюдо 1	Блюдо 2	Блюдо 3	Блюдо 4
User 1	5	4	–	2
User 2	3	–	5	1
User 3	–	5	4	3

Как показано в таблице 1, строки матрицы соответствуют пользователям, столбцы – объектам рекомендаций, а значения отражают взаимодействие пользователя с конкретным объектом. Отсутствующие значения соответствуют объектам, с которыми пользователь ранее не взаимодействовал.

Одним из наиболее известных методов матричной факторизации является метод SVD (Singular Value Decomposition – сингулярное разложение матрицы) [9]. Данный метод широко применяется в рекомендательных системах благодаря способности выявлять скрытые зависимости между пользователями и объектами.

Метод SVD основан на разложении исходной матрицы взаимодействий на несколько матриц меньшей размерности [2]:

$$R \approx U \Sigma V^{-T}, \quad (1)$$

где R – исходная матрица взаимодействий пользователей и объектов;

U – матрица скрытых признаков пользователей;

Σ – диагональная матрица весов факторов;

V^{-T} – матрица скрытых признаков объектов.

Использование данного подхода позволяет выделять скрытые факторы, влияющие на предпочтения пользователей, и прогнозировать вероятность интереса пользователя к объектам, с которыми он ранее не взаимодействовал (рисунок 8).



Рисунок 8 – Общая схема формирования рекомендаций методом SVD

Таким образом, рекомендации формируются не только на основе прямого совпадения заказов пользователей, но и на основе скрытых закономерностей, выявленных моделью в процессе анализа данных [9].

Одной из ключевых особенностей метода SVD является возможность выявления скрытых предпочтений пользователей (рисунок 9). Например, модель может определить, что определённый пользователь предпочитает блюда определённого типа или категории, даже если подобные предпочтения не были явно указаны.

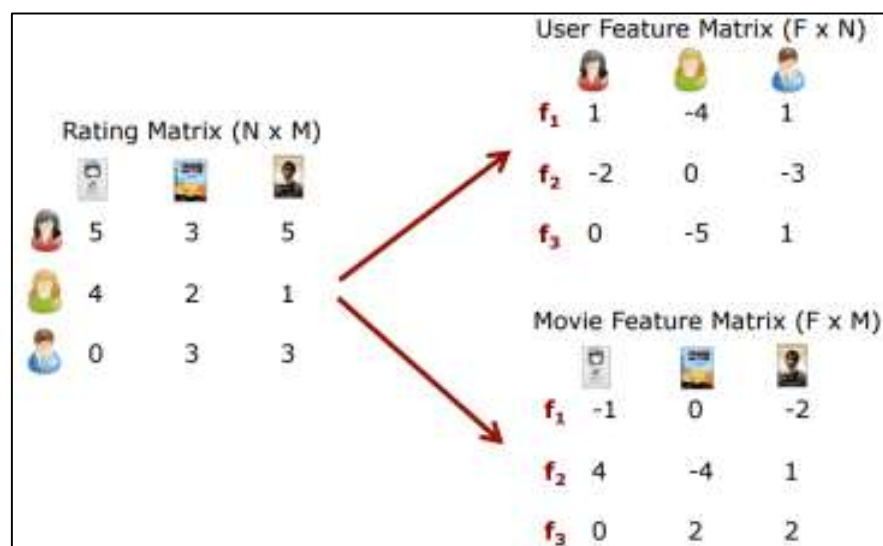


Рисунок 9 – Пример выявления скрытых предпочтений пользователей методом SVD [5]

В рамках разрабатываемой рекомендательной системы метод SVD предполагается использовать для формирования персональных предложений пользователям, имеющим историю заказов. На основе анализа предыдущих взаимодействий система сможет предлагать блюда, которые могут представлять интерес для конкретного клиента.

К основным преимуществам метода SVD можно отнести:

- высокий уровень персонализации рекомендаций,
- способность выявлять скрытые предпочтения пользователей,
- возможность формирования рекомендаций на основе анализа поведения других пользователей,
- высокую эффективность при наличии достаточного объёма данных.

Однако данный метод обладает и рядом ограничений. Одной из основных проблем является проблема холодного старта, возникающая при отсутствии информации о новом пользователе или объекте [1]. Кроме того, качество рекомендаций существенно зависит от объёма накопленных данных о взаимодействии пользователей с системой.

Таким образом, метод SVD является эффективным инструментом формирования персональных рекомендаций при наличии истории взаимодействия пользователей с системой и может быть использован для

реализации блока персональных предложений в разрабатываемой рекомендательной системе.

2.2 Контентно-ориентированные методы

Одним из распространённых подходов к построению рекомендательных систем является content-based фильтрация [1]. В отличие от методов коллаборативной фильтрации, данный подход основывается не на анализе поведения других пользователей, а на анализе характеристик самих объектов рекомендаций.

Основная идея content-based подхода заключается в формировании рекомендаций на основе сходства объектов. Если пользователь проявляет интерес к определённым объектам, система предлагает ему другие объекты со схожими характеристиками [4].

В контексте предприятий общественного питания объектами рекомендаций являются блюда меню. Для формирования рекомендаций система анализирует характеристики блюд и определяет степень их сходства.

К основным характеристикам блюд могут относиться (таблица 2):

- категория блюда,
- тип кухни,
- состав ингредиентов,
- цена,
- наличие определённых тегов или особенностей.

Таблица 2 – Пример характеристик объектов рекомендаций

Блюдо	Категория	Тип кухни	Основные ингредиенты	Цена
Латте	Напиток	Европейская	кофе, молоко	220
Капучино	Напиток	Европейская	кофе, молоко	210
Том Ям	Суп	Азиатская	креветки, специи	450
Чизкейк	Десерт	Европейская	сыр, сливки	260

Как показано в таблице 2, объекты рекомендаций могут описываться набором характеристик, на основе которых определяется степень их сходства. Например, блюда «Латте» и «Капучино» обладают схожими признаками, так как относятся к одной категории и имеют схожий состав ингредиентов.

Для определения степени сходства между объектами в content-based рекомендательных системах часто используется косинусная мера сходства (cosine similarity) [2, 4]. Данный подход позволяет определить, насколько близки объекты друг к другу на основе их признаков.

Формально косинусная мера сходства может быть представлена следующим выражением:

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}, \quad (2)$$

где A и B – векторы признаков объектов;

$A \cdot B$ – скалярное произведение векторов;

$\|A\|$ и $\|B\|$ – длины соответствующих векторов.

Чем ближе значение функции сходства к единице, тем более похожими считаются объекты (рисунок 10).



Рисунок 10 – Пример определения сходства объектов в content-based модели

Использование content-based подхода позволяет формировать рекомендации даже при отсутствии информации о других пользователях системы. Благодаря этому данный метод эффективно решает проблему холодного старта и может использоваться для формирования рекомендаций новым пользователям.

В рамках разрабатываемой рекомендательной системы content-based модель предполагается использовать в нескольких сценариях.

Первым сценарием является формирование рекомендаций на этапе работы с корзиной. В данном случае система анализирует блюда, уже добавленные пользователем в заказ, и предлагает схожие или дополняющие позиции меню. Например, при добавлении кофейного напитка система может предложить десерт или альтернативный напиток со схожими характеристиками.

Вторым сценарием является использование рекомендаций при замене блюда. Если определённая позиция отсутствует или недоступна, система может предложить пользователю похожий объект на основе анализа характеристик блюда.

Третьим сценарием является работа с новыми пользователями. Если пользователь ещё не имеет истории заказов, но указал определённые предпочтения или начал формировать корзину, система может использовать информацию о характеристиках выбранных блюд для формирования рекомендаций (рисунок 11).

К основным преимуществам content-based подхода можно отнести [1, 4]:

- возможность формирования рекомендаций без анализа поведения других пользователей,
- решение проблемы холодного старта,
- простоту интерпретации рекомендаций,
- возможность работы при ограниченном объёме данных.

Однако данный подход обладает и рядом ограничений. В первую очередь, система склонна рекомендовать объекты, максимально похожие на уже выбранные пользователем, что может снижать разнообразие рекомендаций. Кроме того, качество рекомендаций напрямую зависит от полноты и качества описания объектов.



Рисунок 11 – Пример работы content-based рекомендаций в системе

Таким образом, content-based подход является эффективным инструментом формирования рекомендаций на основе характеристик объектов и может использоваться для формирования рекомендаций при работе с корзиной, замене блюд и взаимодействии с новыми пользователями.

2.3 Метод k-ближайших соседей

Одним из распространённых методов построения рекомендательных систем является метод k-ближайших соседей (k-Nearest Neighbors, kNN) [3]. Данный подход широко применяется для поиска схожих объектов и формирования рекомендаций на основе анализа взаимосвязей между ними.

Основная идея метода заключается в том, что объекты со схожими характеристиками или схожим поведением располагаются близко друг к другу в пространстве признаков [3]. В рамках рекомендательных систем это позволяет находить объекты, наиболее близкие к рассматриваемому объекту, и формировать рекомендации на основе найденных соседей.

В рекомендательных системах метод kNN часто применяется в рамках item-based подхода, при котором анализируется сходство между объектами рекомендаций [1]. В отличие от user-based методов, где сравниваются пользователи, item-based подход ориентирован на поиск взаимосвязей между самими объектами.

В контексте предприятий общественного питания объектами рекомендаций являются блюда меню. Анализируя историю заказов пользователей, система может выявлять блюда, которые часто приобретаются совместно.

Например, если пользователи часто добавляют к бургеру картофель фри и соус, система может рекомендовать данные позиции другим пользователям при выборе бургера (таблица 3).

Таблица 3 – Пример совместно приобретаемых блюд

Основное блюдо	Часто приобретаемые позиции
Бургер	Картофель фри, соус
Капучино	Чизкейк
Пицца	Соус, напиток
Суши	Соевый соус, имбирь

Как показано в таблице 3, метод kNN позволяет выявлять взаимосвязи между объектами на основе анализа истории заказов пользователей [4]. Это делает возможным формирование рекомендаций сопутствующих товаров и дополнительных позиций меню.

Принцип работы метода kNN в рамках рекомендательной системы может быть представлен в виде последовательности этапов (рисунок 12):

- анализ истории заказов пользователей,
- поиск объектов, часто приобретаемых совместно,
- определение ближайших соседей,
- формирование рекомендаций сопутствующих товаров.



Рисунок 12 – Общая схема формирования рекомендаций методом kNN

Для определения степени близости объектов метод kNN может использовать различные меры расстояния. Одной из наиболее распространённых является евклидова метрика расстояния [3]:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad , \quad (3)$$

где x и y – векторы признаков объектов;

n – количество признаков;

$d(x, y)$ – расстояние между объектами.

Чем меньше значение расстояния между объектами, тем более близкими считаются объекты в рамках рассматриваемого пространства признаков [3].

В рамках разрабатываемой рекомендательной системы метод kNN предполагается использовать для формирования рекомендаций сопутствующих товаров на этапе оформления заказа. В частности, система сможет предлагать пользователю блюда и дополнительные позиции, которые

часто выбираются совместно с уже добавленными в корзину объектами (рисунок 13).

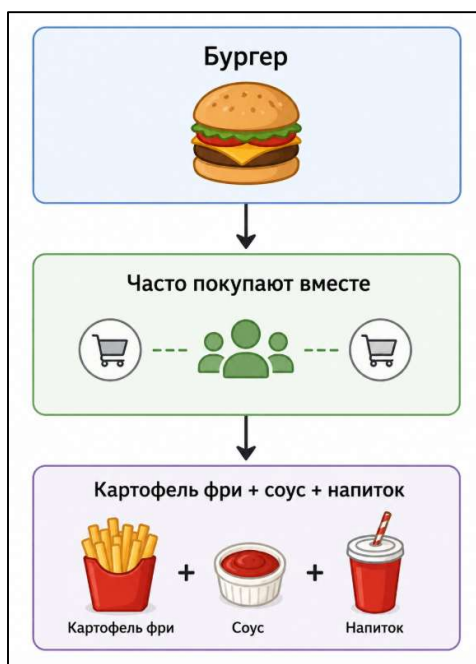


Рисунок 13 – Пример работы рекомендаций сопутствующих товаров методом kNN

К основным преимуществам метода kNN можно отнести [1, 3]:

- простоту реализации,
- высокую интерпретируемость рекомендаций,
- эффективность при формировании рекомендаций сопутствующих товаров,
- возможность повышения среднего чека за счёт допродаж.

Однако данный метод обладает и рядом ограничений. Качество рекомендаций напрямую зависит от объёма накопленных данных о заказах пользователей. При недостаточном количестве данных эффективность рекомендаций может снижаться.

Таким образом, метод k-ближайших соседей является эффективным инструментом формирования рекомендаций сопутствующих товаров и может использоваться для реализации блока дополнительных предложений в процессе оформления заказа.

2.4 Сравнительный анализ методов

Рассмотренные методы построения рекомендательных систем обладают различными особенностями, преимуществами и ограничениями. Для выбора наиболее подходящего подхода к построению рекомендательной системы для предприятий общественного питания малого масштаба необходимо провести сравнительный анализ исследуемых методов.

В рамках данной работы рассматриваются три подхода:

- метод коллаборативной фильтрации на основе SVD,
- content-based фильтрация,
- метод k-ближайших соседей (kNN).

Каждый из указанных методов ориентирован на решение различных задач рекомендательной системы и обладает собственной областью эффективного применения (таблица 4). [4]

Таблица 4 – Сравнение методов построения рекомендательных систем

Критерий	SVD	Content-based	kNN
Уровень персонализации	Высокий	Средний	Низкий
Работа с новыми пользователями	Низкая эффективность	Высокая эффективность	Средняя эффективность
Необходимость истории заказов	Требуется	Не требуется	Частично требуется
Формирование похожих рекомендаций	Ограниченно	Высокая эффективность	Низкая эффективность
Формирование сопутствующих рекомендаций	Ограниченно	Частично	Высокая эффективность
Работа при ограниченном объеме данных	Средняя эффективность	Высокая эффективность	Средняя эффективность
Решение проблемы холодного старта	Нет	Да	Частично
Интерпретируемость рекомендаций	Средняя	Высокая	Высокая
Сложность реализации	Средняя	Низкая	Низкая

Как показано в таблице 4, метод SVD обеспечивает высокий уровень персонализации рекомендаций и позволяет выявлять скрытые предпочтения

пользователей [9]. Благодаря анализу истории взаимодействия пользователей с системой данный подход способен формировать релевантные персональные предложения. Однако эффективность метода существенно снижается при отсутствии данных о пользователе, что связано с проблемой холодного старта [12].

Content-based подход, напротив, не требует информации о поведении других пользователей и способен формировать рекомендации на основе характеристик объектов. Это делает данный метод эффективным при работе с новыми пользователями и в условиях ограниченного объёма данных. Кроме того, content-based фильтрация хорошо подходит для поиска похожих блюд и формирования рекомендаций на основе содержимого корзины пользователя.

Метод k-ближайших соседей демонстрирует высокую эффективность при формировании рекомендаций сопутствующих товаров [4]. Анализируя совместные покупки пользователей, система может выявлять наиболее популярные сочетания блюд и использовать их для формирования дополнительных предложений в процессе оформления заказа.

Несмотря на преимущества каждого из рассмотренных методов, ни один из них не способен полностью решить все задачи рекомендательной системы при самостоятельном использовании [13]. Например, метод SVD плохо подходит для новых пользователей, content-based фильтрация ограничена рекомендацией схожих объектов, а эффективность метода kNN зависит от объёма накопленных данных о заказах.

В связи с этим для разрабатываемой системы целесообразно использовать гибридный подход, основанный на совместном применении нескольких методов рекомендаций [13]. Использование различных моделей для отдельных сценариев взаимодействия с пользователем позволяет повысить качество рекомендаций и обеспечить более гибкую работу системы в условиях ограниченного объёма данных.

Таким образом, результаты сравнительного анализа подтверждают целесообразность совместного использования методов SVD, content-based

фильтрации и k-ближайших соседей в рамках разрабатываемой рекомендательной системы для предприятий общественного питания малого масштаба.

2.5 Выбор моделей для исследования

На основании проведённого сравнительного анализа методов построения рекомендательных систем было принято решение использовать гибридный подход к формированию рекомендаций в рамках разрабатываемой системы для предприятий общественного питания малого масштаба [4, 13].

Использование только одного метода рекомендаций не позволяет эффективно решать все задачи системы. Каждый из рассмотренных подходов обладает собственными преимуществами и ограничениями, вследствие чего применение исключительно одного метода может приводить к снижению качества рекомендаций в отдельных сценариях взаимодействия с пользователем [13].

Метод SVD обеспечивает высокий уровень персонализации рекомендаций и позволяет выявлять скрытые предпочтения пользователей на основе анализа истории заказов [9]. Однако данный подход плохо подходит для новых пользователей, не имеющих истории взаимодействия с системой, поскольку в этом случае отсутствуют необходимые данные для формирования рекомендаций.

Content-based фильтрация, напротив, позволяет формировать рекомендации даже при отсутствии накопленной истории заказов. Данный подход эффективно работает при анализе характеристик блюд и может использоваться для формирования рекомендаций новым пользователям, а также при подборе похожих объектов и альтернативных позиций меню. Однако рекомендации, формируемые content-based моделью, как правило, ограничены объектами со схожими характеристиками и могут обладать недостаточным разнообразием.

Метод k-ближайших соседей демонстрирует высокую эффективность при формировании рекомендаций сопутствующих товаров на основе анализа

совместных покупок пользователей. Вместе с тем данный подход не обеспечивает полноценной персонализации рекомендаций и не позволяет выявлять скрытые предпочтения пользователей.

Таким образом, проведённый анализ показывает, что ни один из рассмотренных методов не способен полностью решить все задачи рекомендательной системы при самостоятельном использовании. В связи с этим целесообразным является использование гибридного подхода, основанного на совместном применении нескольких моделей рекомендаций.

В рамках разрабатываемой системы предполагается распределение методов рекомендаций в зависимости от сценария взаимодействия с пользователем.

Метод SVD предполагается использовать для формирования персональных предложений пользователям, имеющим историю заказов [2]. В данном случае система анализирует предыдущие взаимодействия пользователя с меню и формирует рекомендации на основе выявленных скрытых предпочтений и поведения других пользователей со схожими интересами.

Content-based модель предполагается использовать в сценариях, связанных с анализом характеристик блюд [1]. В частности, данный подход может применяться при формировании рекомендаций новым пользователям, при работе с корзиной пользователя, при подборе похожих блюд, при замене недоступных позиций меню альтернативными объектами.

Метод k-ближайших соседей предполагается использовать для формирования рекомендаций сопутствующих товаров на этапе оформления заказа [4]. Анализируя статистику совместных покупок пользователей, система сможет предлагать дополнительные позиции меню, наиболее часто приобретаемые совместно с выбранными блюдами (рисунок 13).

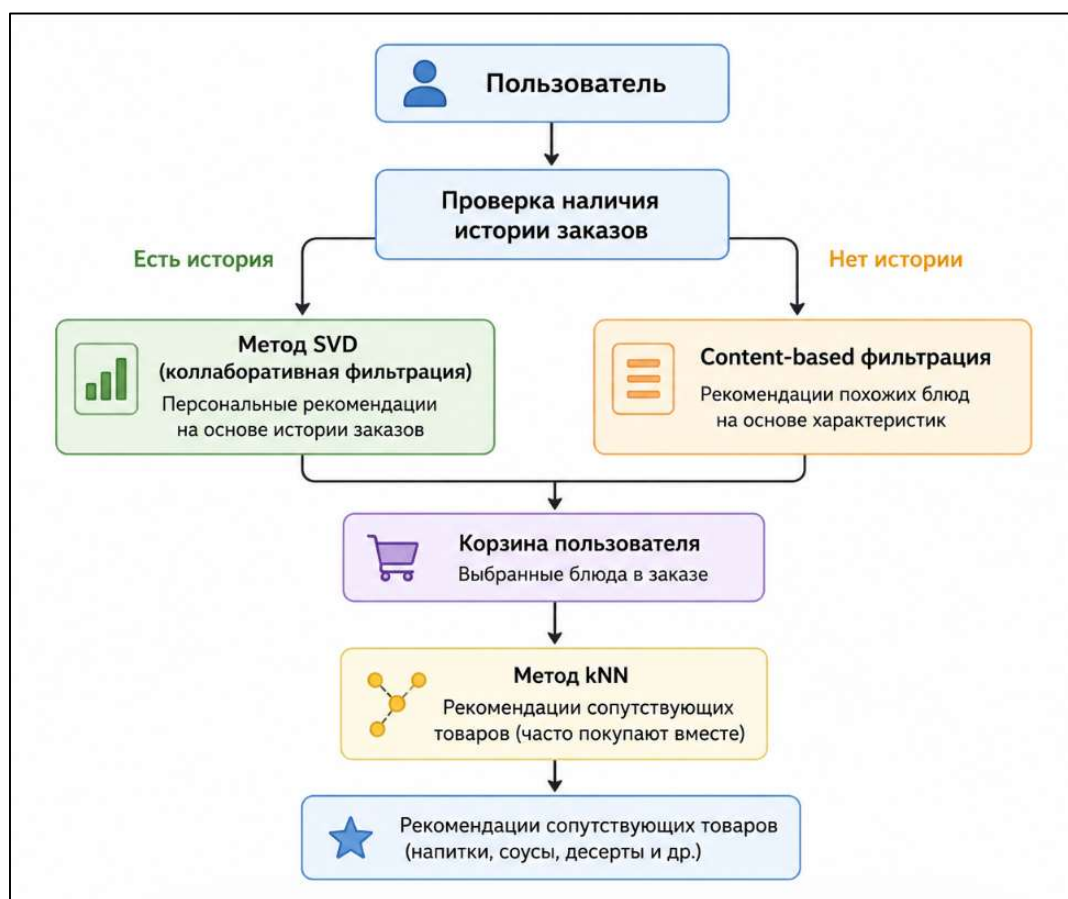


Рисунок 13 – Структура гибридной рекомендательной системы

Как показано на рисунке 13, использование гибридного подхода позволяет распределять методы рекомендаций в зависимости от конкретного сценария работы системы. Такой подход обеспечивает более гибкое формирование рекомендаций и позволяет компенсировать ограничения отдельных моделей.

Использование нескольких методов рекомендаций особенно актуально для предприятий общественного питания малого масштаба, поскольку подобные организации, как правило, обладают ограниченным объёмом накопленных данных и ограниченными вычислительными ресурсами. Применение гибридного подхода позволяет эффективно использовать доступные данные и обеспечивать работу рекомендательной системы даже при недостатке информации о пользователях.

Кроме того, предложенная структура системы позволяет реализовать рекомендации как для постоянных клиентов, так и для новых пользователей. Это особенно важно для предприятий общественного питания, в которых

значительная часть пользователей может не иметь длительной истории взаимодействия с системой.

Использование метода SVD обеспечивает персонализацию рекомендаций для постоянных клиентов, content-based фильтрация позволяет решать проблему холодного старта и формировать рекомендации на основе характеристик блюд, а метод k-ближайших соседей способствует повышению среднего чека за счёт рекомендаций сопутствующих товаров.

Таким образом, использование гибридного подхода, основанного на совместном применении методов SVD, content-based фильтрации и k-ближайших соседей, позволяет учитывать особенности предприятий общественного питания малого масштаба и эффективно решать различные сценарии формирования рекомендаций в рамках разрабатываемой системы.

3. Подготовка данных и обучение моделей

3.1 Источники и структура данных

Для реализации и исследования рекомендательной системы использовались данные существующей информационной системы предприятия общественного питания. Основным источником данных выступила база данных, предназначенная для хранения сведений о клиентах, заказах, позициях меню и составе заказов (рисунок 14).

Использование существующей структуры базы данных позволило обеспечить связь рекомендательной системы с предметной областью предприятия общественного питания и сформировать наборы данных, соответствующие реальным сценариям взаимодействия пользователей с системой.

В рамках исследования из общей структуры базы данных были выделены таблицы, содержащие информацию, необходимую для построения рекомендательной системы:

- таблица `clients`, содержащая сведения о клиентах заведения,
- таблица `orders`, содержащая информацию о заказах пользователей,
- таблица `ordered`, отражающая состав каждого заказа,
- таблица `menu`, содержащая сведения о блюдах и позициях меню.

Таблица `clients` содержит идентификаторы клиентов, сведения о регистрации пользователей и контактную информацию. Данные таблицы используются для идентификации пользователей системы и анализа пользовательской активности.

Таблица `orders` содержит информацию о совершённых заказах, включая идентификатор клиента, дату оформления заказа, статус заказа и итоговую стоимость. Данные таблицы позволяют анализировать историю взаимодействия пользователей с системой.

Таблица `ordered` является связующей таблицей между заказами и позициями меню и содержит перечень блюд, входящих в состав заказа. Данная таблица представляет особую значимость для рекомендательной системы,

поскольку позволяет анализировать совместные покупки пользователей и формировать рекомендации сопутствующих товаров.

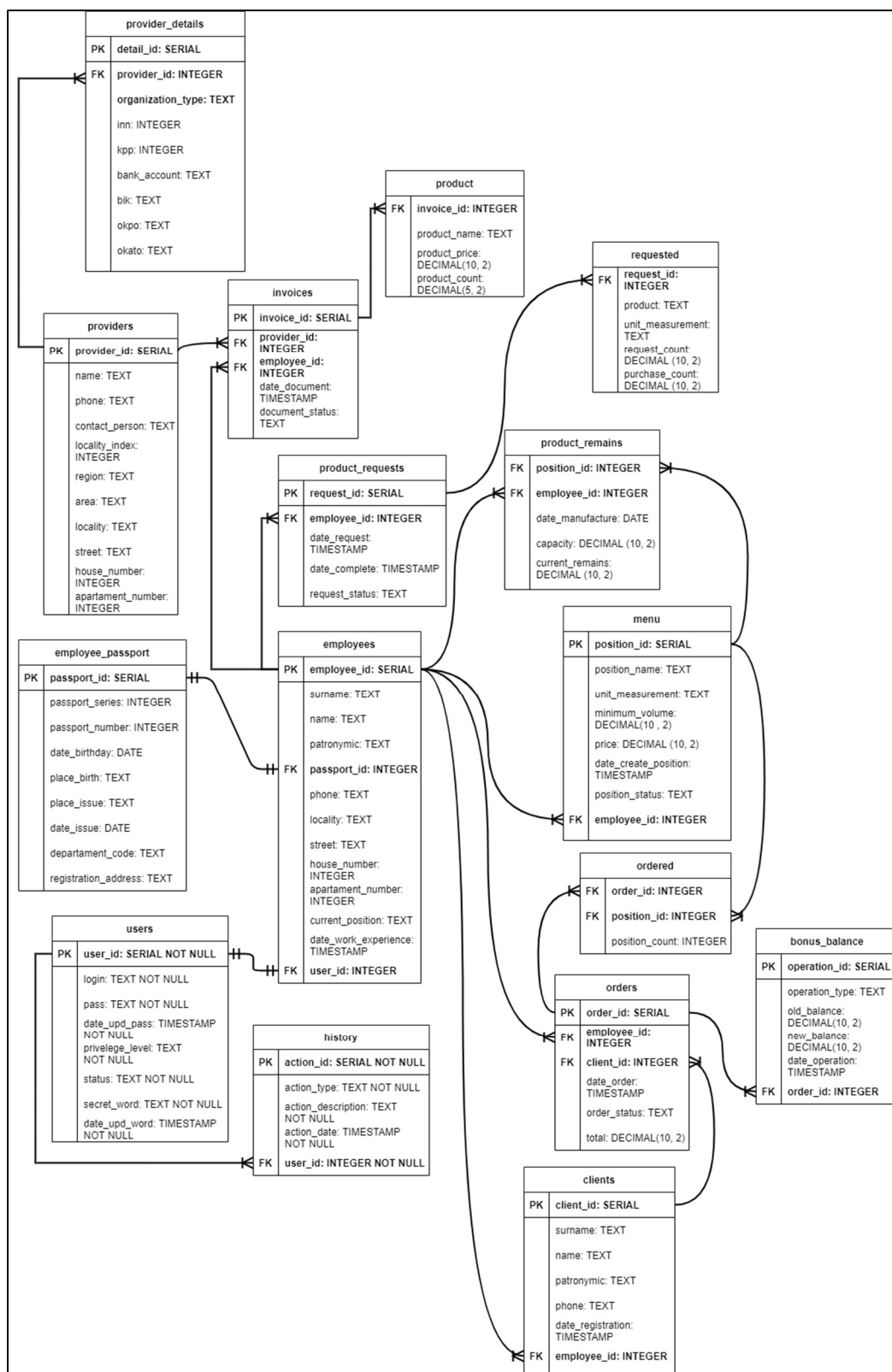


Рисунок 14 – Общая структура базы данных информационной системы предприятия общественного питания

Таблица menu содержит сведения о позициях меню, включая название блюда, стоимость, единицу измерения и статус позиции. Информация из

данной таблицы используется для формирования признаков объектов рекомендаций и построения content-based модели (таблица 5).

Таблица 5 – Использование таблиц базы данных в рекомендательной системе

Таблица	Назначение в рекомендательной системе
clients	Идентификация пользователей
orders	Анализ истории заказов
ordered	Анализ состава заказов и совместных покупок
menu	Формирование признаков объектов рекомендаций

Несмотря на использование существующей базы данных предприятия общественного питания, в рамках исследования возникли ограничения, связанные со спецификой предприятий малого масштаба.

Для небольших заведений общественного питания характерны:

- ограниченное количество клиентов,
- сравнительно небольшой объём накопленных заказов,
- отсутствие обязательной регистрации пользователей,
- неполнота пользовательских данных.

Указанные особенности затрудняют использование исключительно реальных данных для полноценного обучения и тестирования рекомендательных моделей. В частности, ограниченный объём пользовательских взаимодействий снижает эффективность методов коллаборативной фильтрации и усложняет оценку качества рекомендаций.

В связи с этим в рамках исследования использовался комбинированный подход к формированию данных. Часть информации была получена из существующей структуры базы данных, а часть данных была синтезирована искусственно на основе типовых сценариев поведения пользователей и структуры предметной области.

Синтетические данные формировались с учётом:

- популярных сочетаний блюд,
- повторяющихся пользовательских заказов,
- категорий блюд,
- типовых пользовательских предпочтений,

– наиболее распространенных сценариев оформления заказов.

Использование синтетических данных позволило увеличить объём пользовательских взаимодействий и обеспечить достаточное количество данных для проведения экспериментов с рекомендательными моделями.

При этом синтетические данные не заменяли реальные сведения полностью, а использовались для расширения существующего набора данных и моделирования поведения пользователей в условиях ограниченного объёма информации.

На основе существующей структуры базы данных и сформированных синтетических данных были подготовлены специализированные наборы данных, предназначенные для работы отдельных рекомендательных моделей.

Для реализации content-based модели был сформирован набор данных menu_dataset, содержащий характеристики блюд и признаки объектов рекомендаций.

Для реализации метода k-ближайших соседей был сформирован набор данных orders_dataset, содержащий сведения о заказах пользователей и совместных покупках блюд.

Для реализации модели SVD был сформирован набор данных ratings_dataset, представляющий собой матрицу взаимодействий пользователей и объектов рекомендаций (таблица 6).

Таблица 6 – Наборы данных, используемые в работе

Набор данных	Назначение	Используемая модель
menu_dataset	Характеристики блюд и признаки объектов	Content-based
orders_dataset	История заказов и совместные покупки	kNN
ratings_dataset	Матрица пользователь-объект	SVD

Таким образом, использование существующей структуры базы данных предприятия общественного питания в сочетании с синтетически сформированными данными позволило подготовить наборы данных,

необходимые для реализации и исследования гибридной рекомендательной системы.

После формирования наборов данных была выполнена их предварительная обработка и подготовка к использованию в рекомендательных моделях.

3.2 Предобработка данных

После формирования наборов данных была выполнена их предварительная обработка и подготовка к использованию в рекомендательных моделях. Предобработка данных включала загрузку данных в среду разработки, проверку корректности записей, анализ структуры наборов данных, формирование признаков объектов и подготовку данных для алгоритмов машинного обучения.

Для реализации предобработки данных использовалась среда Google Colab и язык программирования Python. Основными библиотеками, применяемыми в процессе обработки данных, выступили pandas, numpy, matplotlib и scikit-learn.

На первом этапе была выполнена загрузка подготовленных CSV-файлов в память программной среды. Для дальнейшей работы использовались три набора данных: menu_dataset.csv, orders_dataset.csv, ratings_dataset.csv.

Ниже представлен фрагмент кода загрузки наборов данных:

```
menu_df = pd.read_csv("menu_dataset_cafe_100.csv")
orders_df = pd.read_csv("orders_dataset_cafe.csv")
ratings_df = pd.read_csv("ratings_dataset_cafe.csv")
```

После загрузки была выполнена проверка структуры и размерности наборов данных (таблица 7).

Таблица 7 – Основные характеристики наборов данных

Набор данных	Количество строк	Количество признаков
menu_dataset	100	7
orders_dataset	5211	8
ratings_dataset	3776	3

Для проверки корректности загрузки данных также был выполнен вывод первых строк наборов данных (рисунок 15).

	item_id	name	category	cuisine	price	ingredients	tags
0	1	Борщ	Первое блюдо	Русская	180	свекла, капуста, картофель, говядина	суп, горячее, мясное
1	2	Щи из свежей капусты	Первое блюдо	Русская	170	капуста, картофель, морковь, говядина	суп, горячее, мясное
2	3	Куриный суп с лапшой	Первое блюдо	Русская	160	курица, лапша, морковь, зелень	суп, горячее, курица
3	4	Рассольник	Первое блюдо	Русская	175	перловка, огурцы, картофель, мясо	суп, горячее, кислое
4	5	Солянка	Первое блюдо	Русская	220	мясо, колбаса, огурцы, лимон	суп, горячее, мясное

	order_id	user_id	item_id	item_name	category	quantity	order_date	order_total
0	1	99	2	Щи из свежей капусты	Первое блюдо	2	2025-01-16	1520
1	1	99	37	Греческий салат	Салат	2	2025-01-16	1520
2	1	99	14	Глов с курицей	Второе блюдо	2	2025-01-16	1520
3	1	99	79	Сок апельсиновый	Напиток	1	2025-01-16	1520
4	2	7	10	Окрошка	Первое блюдо	2	2025-02-17	810

	user_id	item_id	rating
0	1	1	1
1	1	4	2
2	1	8	1
3	1	9	2

Рисунок 15 – Пример загруженных наборов данных

После загрузки наборов данных была выполнена проверка качества данных. На данном этапе выполнялся анализ пропущенных значений, дубликатов, некорректных записей и типов данных.

Проверка пропущенных значений показала отсутствие незаполненных полей в используемых наборах данных.

Ниже представлен фрагмент кода проверки пропущенных значений:

```
menu_df.isnull().sum()
orders_df.isnull().sum()
ratings_df.isnull().sum()
```

Для проверки дубликатов записей использовался следующий код:

```
menu_df.duplicated().sum()
orders_df.duplicated().sum()
ratings_df.duplicated().sum()
```

В результате проверки полных дубликатов записей обнаружено не было.

Дополнительно была выполнена проверка корректности числовых значений, включая анализ отрицательных цен, некорректного количества позиций в заказах и рейтингов вне допустимого диапазона:

```
invalid_prices = menu_df[menu_df["price"] <= 0]
invalid_quantity = orders_df[orders_df["quantity"] <= 0]
```

```
invalid_ratings = ratings_df[
    (ratings_df["rating"] < 1) |
    (ratings_df["rating"] > 5)
]
```

В ходе проверки некорректных значений обнаружено не было. Полученные результаты позволили использовать наборы данных без дополнительного восстановления или очистки записей.

Для реализации content-based модели был сформирован объединённый текстовый признак features, содержащий основные характеристики блюд.

В состав признака features были включены:

- категория блюда,
- тип кухни,
- используемые ингредиенты,
- тематические теги.

Формирование объединённого признака позволило представить характеристики блюд в едином текстовом виде и использовать их для последующего анализа сходства объектов (таблица 8).

Ниже представлен фрагмент кода формирования объединённого признака:

```
menu_df["features"] = (
    menu_df["category"].astype(str) + " " +
    menu_df["cuisine"].astype(str) + " " +
    menu_df["ingredients"].astype(str) + " " +
    menu_df["tags"].astype(str)
)
```

Таблица 8 – Пример сформированного признака features

Блюдо	features
Борщ	Первое блюдо Русская свекла капуста картофель мясное суп горячее
Капучино	Кофе Европейская кофе молоко горячее молочное
Пирожок с мясом	Выпечка Русская тесто фарш мясное сытное

После формирования текстового признака *features* была выполнена TF-IDF-векторизация характеристик блюд [2].

Метод TF-IDF позволяет преобразовать текстовое описание объектов в числовое представление, отражающее значимость отдельных терминов в рамках набора данных.

$$TF-IDF(t, d) = TF(t, d) \cdot \log \frac{N}{DF(t)}, \quad (3)$$

где $TF(t, d)$ – частота термина t в документе d ;

$DF(t)$ – количество документов, содержащих термин,

N – общее количество документов.

Ниже представлен фрагмент кода, реализующий метод TF-IDF:

```
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(
    menu_df["features"]
)
```

В результате выполнения TF-IDF-векторизации каждое блюдо было представлено в виде числового вектора признаков, используемого для расчёта сходства объектов в content-based модели.

Для реализации модели SVD использовался набор данных *ratings_dataset*, содержащий информацию о взаимодействиях пользователей и объектов рекомендаций.

Поскольку в информационной системе отсутствовали явные пользовательские оценки блюд, в качестве рейтинга использовалась частота заказов конкретного блюда пользователем.

Частота заказов преобразовывалась в условную шкалу предпочтений от 1 до 5 по следующему правилу:

$$rating = \min(order_count, 5), \quad (4)$$

где $order_count$ – количество заказов блюда пользователем;

$\min(order_count, 5)$ ограничивает максимальное значение рейтинга числом 5;

rating – итоговая пользовательская оценка.

Ниже представлен фрагмент кода, преобразовывающий условную шкалу предпочтений по заданному правилу:

```
ratings_df = (  
    orders_df.groupby(  
        ["user_id", "item_id"]  
    )["quantity"]  
    .sum()  
    .reset_index()  
)
```

После формирования пользовательских рейтингов был выполнен анализ структуры матрицы взаимодействий пользователь-объект.

Матрица взаимодействий является разреженной, поскольку каждый пользователь взаимодействует только с ограниченным количеством объектов рекомендаций.

Процесс формирования матрицы представлен ниже:

```
possible_interactions = num_users * num_items  
actual_interactions = len(ratings_df)  
  
sparsity = 1 - (  
    actual_interactions /  
    possible_interactions  
)
```

В результате расчёта была определена степень разреженности пользовательской матрицы взаимодействий.

Дополнительно был выполнен анализ распределения пользовательских рейтингов и категорий блюд.

Анализ распределения рейтингов позволяет оценить структуру пользовательских взаимодействий и выявить наиболее распространённые значения рейтингов. (рисунок 16)

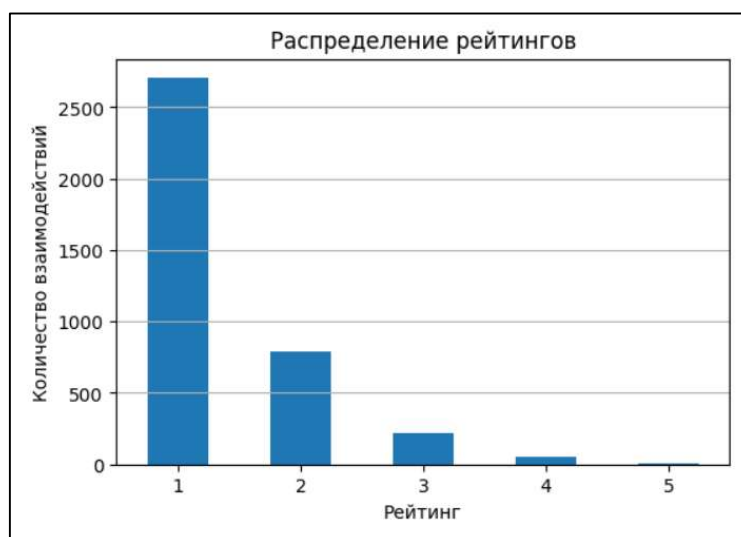


Рисунок 16 – Распределение пользовательских рейтингов

Также был выполнен анализ распределения заказов по категориям блюд (рисунок 17).

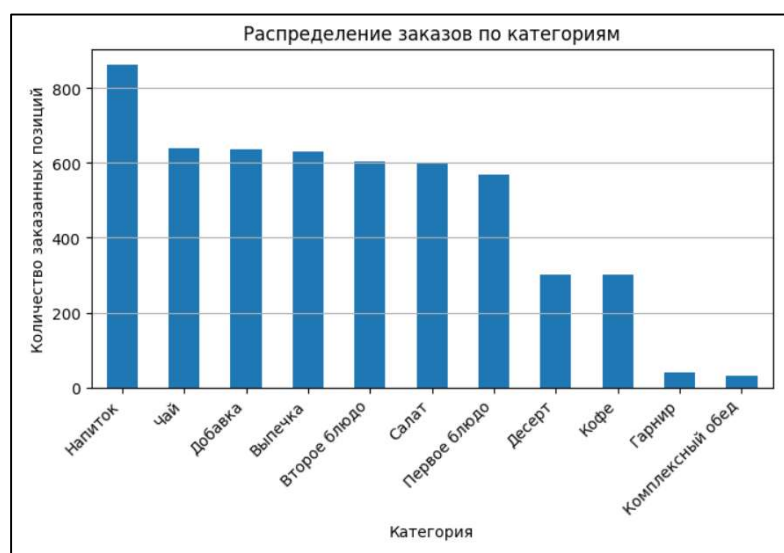


Рисунок 17 – Распределение заказов по категориям блюд

Полученные результаты показывают, что различные категории блюд используются пользователями с разной частотой, что создаёт необходимые условия для формирования персональных рекомендаций.

Таким образом, в рамках предобработки данных были выполнены загрузка и проверка наборов данных, анализ корректности записей, формирование признаков объектов, подготовка пользовательских рейтингов и преобразование текстовых характеристик в числовое представление. Полученные результаты были использованы для дальнейшего обучения и исследования рекомендательных моделей.

3.3 Формирование пользовательских предпочтений

После выполнения предварительной обработки данных был реализован этап формирования пользовательских предпочтений. Основной целью данного этапа являлось определение интересов пользователей на основе истории их взаимодействий с системой.

Формирование пользовательских предпочтений представляет собой важный этап построения рекомендательной системы, поскольку позволяет определить особенности поведения пользователей и использовать полученную информацию при генерации персональных рекомендаций.

В рамках исследования пользовательские предпочтения формировались на основе следующих данных:

- истории заказов пользователей,
- частоты взаимодействий с блюдами,
- популярных категорий блюд,
- совместных покупок,
- общего уровня пользовательской активности.

В сформированном наборе данных пользователи обладают различным уровнем активности. Часть пользователей имеет ограниченную историю взаимодействий с системой, тогда как другая часть активно использует сервис и формирует большое количество заказов. Подобное распределение позволяет приблизить структуру данных к реальным условиям функционирования рекомендательных систем.

На первом этапе был выполнен анализ активности пользователей. Для каждого пользователя определялись:

- количество оформленных заказов,
- общее количество заказанных позиций,
- количество уникальных блюд, с которыми взаимодействовал пользователь.

Полученные характеристики позволяют оценить объём пользовательской истории и определить степень активности пользователей в системе.

Ниже представлен фрагмент кода формирования пользовательской активности:

```
user_activity = (
    orders_df.groupby("user_id")
    .agg(
        orders_count=("order_id", "nunique"),
        items_count=("quantity", "sum"),
        unique_items=("item_id", "nunique")
    )
    .reset_index()
)
```

В результате выполнения расчётов была сформирована таблица пользовательской активности (рисунок 18).

Пользовательская активность успешно рассчитана.
Количество пользователей: 100

	user_id	orders_count	items_count	unique_items
0	1	1	3	2
1	2	1	5	3
2	3	2	10	6
3	4	1	4	4
4	5	1	2	2
5	6	1	2	2
6	7	1	3	3
7	8	1	5	3
8	9	2	7	4
9	10	1	4	2

Рисунок 18 – Получение пользовательской активности

Следующим этапом было определение наиболее популярных категорий блюд для каждого пользователя. Для этого выполнялась группировка истории заказов пользователей по категориям блюд с последующим подсчётом количества взаимодействий.

Реализация определения популярных категорий блюд приведена ниже:

```
user_category_preferences = (
```

```

orders_df
.groupby(["user_id", "category"])
.agg(
    category_quantity=("quantity", "sum")
)
.reset_index()
)

```

После расчёта предпочтений для каждого пользователя были определены наиболее часто встречающиеся категории блюд (рисунок 19).

Полученные данные позволяют определить основные интересы пользователей и использовать их при построении рекомендаций.

Предпочтения пользователей по категориям успешно рассчитаны.				
	user_id	category	category_quantity	category_orders
0	1	Выпечка	2	1
1	1	Кофе	1	1
2	2	Выпечка	2	1
4	2	Чай	2	1
3	2	Добавка	1	1
5	3	Выпечка	4	2
6	3	Добавка	3	2
7	3	Чай	3	2
8	4	Второе блюдо	1	1
9	4	Напиток	1	1

Рисунок 19 – Предпочтения пользователей по категориям блюд

Полученные результаты показывают, что пользователи обладают устойчивыми предпочтениями по отношению к определённым категориям блюд. Это позволяет использовать историю взаимодействий в качестве основы для формирования персональных рекомендаций.

Помимо категорий блюд были определены наиболее часто заказываемые блюда каждого пользователя. Для этого выполнялся анализ количества заказов конкретных позиций меню пользователями системы.

Ниже представлен фрагмент кода, определяющий часто заказываемые блюда:

```

user_item_preferences = (
    orders_df

```

```

        .groupby(["user_id", "item_id", "item_name"])
        .agg(
            item_quantity=("quantity", "sum")
        )
        .reset_index()
    )

```

После обработки данных для каждого пользователя были определены наиболее часто заказываемые блюда.

Полученные сведения используются при построении персональных рекомендаций и анализе пользовательских интересов (рисунок 20).

Топ-5 блюд для каждого пользователя сформирован.		
index	user_id	favorite_items
0	1	Самса с курицей, Раф
1	2	Булочка с корицей, Чай зеленый, Хлеб черный
2	3	Ватрушка с творогом, Пирог с ягодами, Чай имбирный, Майонез, Чай черный
3	4	Куриный суп с лапшой, Овощное рагу, Морковь по-корейски, Минеральная вода
4	5	Самса с курицей, Мокко

Рисунок 20 – Предпочтения пользователей по конкретным блюдам

На основе рассчитанных характеристик был сформирован пользовательский профиль, объединяющий сведения об активности пользователя, любимых категориях блюд и наиболее популярных позициях меню.

Пользовательский профиль представляет собой агрегированное описание интересов пользователя и используется при дальнейшей работе рекомендательных моделей.

Программная реализация представлена ниже:

```

user_profile = user_activity.merge(
    top_categories,
    on="user_id",
    how="left"
)

user_profile = user_profile.merge(
    top_items,
    on="user_id",
    how="left"
)

```

В результате выполнения обработки был сформирован агрегированный профиль каждого пользователя системы (рисунок 21).

Профиль пользователя дополнен любимыми блюдами.

	user_id	orders_count	items_count	unique_items	favorite_categories	favorite_items
0	1	1	3	2	Выпечка, Кофе	Самса с курицей, Раф
1	2	1	5	3	Выпечка, Чай, Добавка	Булочка с корицей, Чай зеленый, Хлеб черный
2	3	2	10	6	Выпечка, Добавка, Чай	Ватрушка с творогом, Пирог с ягодами, Чай имби...
3	4	1	4	4	Второе блюдо, Напиток, Первое блюдо	Куриный суп с лапшой, Овощное рагу, Морковь по...
4	5	1	2	2	Выпечка, Кофе	Самса с курицей, Мокко
5	6	1	2	2	Напиток, Салат	Цезарь с курицей, Компот из сухофруктов
6	7	1	3	3	Выпечка, Добавка, Чай	Хачапури, Чай ягодный, Соус чесночный
7	8	1	5	3	Выпечка, Кофе	Пирожок с картошкой, Пирожок с мясом, Айс-латте
8	9	2	7	4	Комплексный обед, Кофе, Гарнир	Обед с рыбой, Кофе с сиропом, Картофель по-дер...
9	10	1	4	2	Напиток, Салат	Сельдь под шубой, Компот из сухофруктов

Рисунок 21 – Агрегированный пользовательский профиль

Дополнительно была выполнена классификация пользователей по объёму истории заказов.

Пользователи были разделены на новых пользователей, пользователей с ограниченной историей взаимодействий и активных пользователей (рисунок 22).

Использование пользователей с различным объёмом истории взаимодействий позволяет моделировать ситуацию холодного старта и исследовать работу рекомендательной системы в условиях неполноты пользовательских данных.

Функция распределения пользователей по группам приведен ниже:

```
def define_user_group(orders_count):

    if orders_count <= 2:
        return "Новый пользователь"

    elif orders_count <= 5:
        return "Пользователь с малой историей"

    else:
        return "Активный пользователь"
```

Распределение пользователей по группам:

	user_group	users_count
0	Активный пользователь	50
1	Пользователь с малой историей	30
2	Новый пользователь	20

Рисунок 22 – Распределение пользователей по уровню активности

Полученные результаты показывают наличие пользователей с различным уровнем активности, что позволяет исследовать работу рекомендательной системы в условиях неоднородности пользовательских данных.

Для подготовки данных к использованию метода k-ближайших соседей был выполнен анализ совместных покупок.

На основе состава каждого заказа формировались пары блюд, совместно приобретаемых пользователями. Далее для каждой пары рассчитывалось количество совместных появлений в заказах.

Алгоритм формирования наиболее частых совместных покупок приведен ниже:

```
for basket in order_baskets["item_name"]:  
  
    unique_items = sorted(set(basket))  
  
    for item_pair in combinations(unique_items, 2):  
        pair_counter[item_pair] += 1
```

В результате выполнения анализа был сформирован набор данных совместных покупок (рисунок 23), используемый при построении рекомендаций сопутствующих товаров.

Топ-10 наиболее частых совместных покупок:			
	item_1	item_2	purchase_count
0	Компот из сухофруктов	Цезарь с курицей	9
1	Борщ	Сок яблочный	9
2	Макароны с сыром	Сок яблочный	9
3	Сок апельсиновый	Суп-пюре тыквенный	8
4	Лимонад	Солянка	8
5	Горчица	Сок апельсиновый	8
6	Гуляш с макаронами	Морс клюквенный	8
7	Минеральная вода	Солянка	8
8	Греческий салат	Сок яблочный	8
9	Жаркое по-домашнему	Минеральная вода	8

Рисунок 23 – Набор совместных покупок

Полученные результаты показывают наличие устойчивых совместных покупок между отдельными позициями меню. Выявленные закономерности могут использоваться при формировании рекомендаций сопутствующих товаров и комплексных предложений.

Для демонстрации результатов формирования пользовательских предпочтений был рассмотрен пример пользовательского профиля, содержащего основные характеристики взаимодействия пользователя с системой (таблица 9).

Таблица 9 – Пример пользовательского профиля

Параметр	Значение
Идентификатор пользователя	15
Количество заказов	4
Количество заказанных позиций	11
Количество уникальных блюд	9
Любимые категории	Кофе, Выпечка
Любимые блюда	Капучино, Круассан
Группа пользователя	Пользователь с малой историей

Таким образом, в рамках этапа формирования пользовательских предпочтений были определены основные характеристики пользовательского поведения, сформированы пользовательские профили и рассчитаны сведения о совместных покупках блюд. Полученные результаты используются при дальнейшей реализации и исследовании рекомендательных моделей.

3.4 Реализация и обучение рекомендательных моделей

3.4.1 Обучение модели коллаборативной фильтрации на основе SVD

Для построения модели коллаборативной фильтрации использовался ранее подготовленный набор рейтингов пользователей, содержащий информацию о взаимодействиях пользователей с блюдами меню. В качестве входных данных модели применялись идентификаторы пользователей, идентификаторы блюд и соответствующие рейтинги взаимодействий.

Для подготовки данных к обучению использовалась библиотека Surprise [11], предназначенная для построения рекомендательных систем на основе коллаборативной фильтрации. На первом этапе данные были преобразованы

во внутренний формат библиотеки с указанием диапазона возможных рейтингов.

Ниже представлен фрагмент кода, формирующий диапазон возможных рейтингов:

```
data = Dataset.load_from_df(
    ratings_df[["user_id", "item_id", "rating"]],
    reader
)
```

После подготовки данных выполнялось разделение выборки на обучающую и тестовую части. Обучающая выборка использовалась для настройки параметров модели, а тестовая - для получения предсказаний и последующей оценки качества рекомендаций.

Программная реализация представлена ниже:

```
trainset, testset = train_test_split(
    data,
    test_size=0.2,
    random_state=42
)
```

Для реализации модели использовался алгоритм сингулярного разложения матриц SVD, позволяющий выявлять скрытые пользовательские предпочтения на основе истории взаимодействий [11]. Использование SVD-модели позволяет выявлять скрытые факторы пользовательских предпочтений, напрямую не представленные в исходных данных. При создании модели задавались параметры количества скрытых факторов и числа эпох обучения.

Для реализации данного этапа использовался следующий фрагмент кода:

```
svd_model = SVD(
    n_factors=50,
    n_epochs=30,
    random_state=42
)

svd_model.fit(trainset)
```

После обучения модели выполнялось получение предсказанных рейтингов для тестовой выборки. Модель прогнозировала степень потенциального интереса пользователя к определённым позициям меню.

Ниже показан пример реализации на языке Python:

```

predicted_rating = svd_model.predict(
    user_id,
    item_id
).est

```

На основе предсказанных рейтингов была реализована функция формирования персональных рекомендаций. Для каждого пользователя из рекомендаций исключались блюда, с которыми ранее уже наблюдались взаимодействия, после чего выполнялась сортировка объектов по величине предсказанного рейтинга и формирование списка Топ-N рекомендаций.

В отличие от content-based модели и модели сопутствующих товаров kNN, SVD-модель формирует рекомендации с учётом индивидуальных особенностей поведения пользователей.

Для проверки корректности работы модели было выполнено тестирование на нескольких пользователях из датасета. В процессе тестирования модель формировала персональные рекомендации, основанные на скрытых паттернах пользовательского поведения и истории взаимодействий. Качество рекомендаций модели в значительной степени зависит от объёма и полноты пользовательских взаимодействий, используемых в процессе обучения (таблица 10).

Таблица 10 – Пример рекомендаций SVD-модели

Пользователь	Рекомендуемое блюдо	Predicted rating
84	Раф	2.284
84	Самса с курицей	2.267
61	Лимонад	1.771
61	Круассан	1.742

Полученные результаты демонстрируют возможность применения модели SVD для формирования персональных рекомендаций на основе пользовательских предпочтений и истории взаимодействий с объектами меню. Полученные результаты и качество рекомендаций будут дополнительно проанализированы в главе 4.

3.4.2 Построение content-based модели рекомендаций

Для построения content-based модели рекомендаций использовались ранее сформированные признаки блюд. В качестве основы модели

применялась TF-IDF матрица признаков и матрица сходства объектов, рассчитанная на основе cosine similarity [2]. Использование данного подхода позволило формировать рекомендации на основе характеристик блюд без необходимости анализа истории взаимодействий других пользователей.

После формирования матрицы сходства была реализована функция получения рекомендаций для выбранного блюда. Для обеспечения быстрого доступа к объектам меню была выполнена индексация блюд по их названиям. На основе сформированных индексов осуществлялся поиск соответствующего объекта в матрице сходства и определение наиболее похожих позиций меню.

Реализация данного функционала представлена следующим кодом:

```
indices = pd.Series(  
    menu_df.index,  
    index=menu_df["name"]  
) .drop_duplicates()
```

Для формирования рекомендаций была реализована функция `get_similar_items()`, выполняющая поиск наиболее похожих блюд на основе коэффициентов сходства. Функция принимает на вход название блюда и количество рекомендуемых позиций, после чего возвращает список наиболее близких объектов меню.

Реализация описанного алгоритма представлена следующим образом:

```
def get_similar_items(item_name, top_n=5):  
  
    item_idx = indices[item_name]  
  
    similarity_scores = list(  
        enumerate(cosine_sim[item_idx])  
    )  
  
    similarity_scores = sorted(  
        similarity_scores,  
        key=lambda x: x[1],  
        reverse=True  
    )  
  
    similarity_scores = similarity_scores[1:top_n + 1]
```

После реализации функции рекомендаций было выполнено тестирование модели на различных позициях меню. Для повышения

объективности тестирования набор блюд для проверки формировался случайным образом из существующего датасета меню.

Программная реализация представлена ниже:

```
test_items = menu_df["name"].sample(  
    n=5,  
    random_state=42  
) .tolist()
```

В процессе тестирования для каждого выбранного блюда формировался список наиболее похожих позиций меню. Полученные рекомендации демонстрировали логическую согласованность и соответствовали характеристикам исходных объектов. Например, для кофейных напитков модель предлагала другие напитки аналогичной категории и состава, а для десертов – схожие сладкие блюда (таблица 11).

Таблица 11 – Пример рекомендаций content-based модели

Исходное блюдо	Рекомендуемые блюда
Капучино	Латте, Раф, Айс-латте
Чизкейк	Медовик, Маффин, Сырники
Борщ	Солянка, Крем-суп, Лагман

В результате была реализована content-based модель рекомендаций, способная формировать рекомендации похожих блюд на основе характеристик объектов меню. Реализованный подход может использоваться в сценариях холодного старта, а также при формировании рекомендаций на основе текущего выбора пользователя.

3.4.3 Реализация модели поиска товаров методом kNN

Для построения модели поиска сопутствующих товаров использовались данные совместных заказов, подготовленные ранее. В качестве основы модели применялась матрица взаимодействий «заказ - блюдо», содержащая информацию о совместном появлении позиций меню в пользовательских заказах.

После подготовки матрицы взаимодействий было выполнено её транспонирование для формирования item-item представления данных, в

котором каждая строка соответствовала отдельному блюду, а признаки отражали информацию о совместных покупках с другими позициями меню.

Ниже приведён фрагмент программного кода:

```
item_order_matrix = order_item_matrix.T
```

Для реализации модели поиска ближайших объектов использовался алгоритм NearestNeighbors библиотеки scikit-learn [10]. В качестве метрики близости применялась cosine similarity, позволяющая оценивать степень взаимосвязи между блюдами на основе их совместного присутствия в заказах пользователей.

Пример реализации модели представлен ниже:

```
knn_model = NearestNeighbors(  
    metric="cosine",  
    algorithm="brute"  
)  
  
knn_model.fit(item_order_matrix)
```

После обучения модели была реализована функция `get_knn_recommendations()`, выполняющая поиск ближайших соседей для выбранного блюда. Функция формирует список сопутствующих товаров, которые наиболее часто встречаются в заказах совместно с исходной позицией меню.

Для выполнения операции использовался следующий программный код:

```
def get_knn_recommendations(item_name, top_n=5):  
  
    item_vector = item_order_matrix.loc[[item_name]]  
  
    distances, indices = knn_model.kneighbors(  
        item_vector,  
        n_neighbors=top_n + 1  
    )
```

Для проверки корректности работы модели было выполнено тестирование на случайно выбранных блюдах из датасета заказов. Для каждого объекта формировался список наиболее близких позиций меню на основе истории совместных покупок.

Программная реализация представлена ниже:

```
test_items_knn = item_order_matrix.index.to_series().sample(  
    n=5,
```

```

random_state=42
).tolist()

```

В процессе тестирования модель формировала рекомендации сопутствующих товаров, соответствующие типичным сценариям заказов пользователей. Например, для напитков система предлагала десерты, выпечку и дополнительные напитки, а для основных блюд – гарниры, салаты и другие позиции меню, часто приобретаемые совместно (рисунок 24).

Результаты тестирования kNN-модели:			
	source_item	recommended_item	similarity
0	Рыба с рисом	Грибной суп	0.138
1	Рыба с рисом	Куриный суп с лапшой	0.123
2	Рыба с рисом	Борщ	0.116
3	Рыба с рисом	Морковь по-корейски	0.101
4	Рыба с рисом	Греческий салат	0.099
5	Медовик	Лимон	0.231
6	Медовик	Гуляш с макаронами	0.227
7	Медовик	Суп-пюре тыквенный	0.201
8	Медовик	Лимонад	0.140
9	Медовик	Плюшка с сахаром	0.110
10	Чайный набор	Латте	0.166
11	Чайный набор	Мокко	0.101
12	Чайный набор	Кофе с сиропом	0.086

Рисунок 24 – Пример рекомендаций kNN-модели

Полученные результаты демонстрируют возможность применения модели kNN для формирования рекомендаций сопутствующих товаров в рамках рекомендательной системы заведений общественного питания.

3.4.4 Формирование гибридной рекомендательной системы

В процессе разработки рекомендательной системы были реализованы несколько моделей, использующих различные подходы к формированию рекомендаций. Каждая из моделей обладает определёнными преимуществами и ограничениями, что затрудняет использование только одного подхода для решения всех задач рекомендательной системы.

Content-based модель эффективно формирует рекомендации похожих блюд на основе характеристик объектов меню и может использоваться в

сценариях холодного старта. Однако данный подход не учитывает особенности пользовательского поведения и историю взаимодействий пользователей с системой.

Модель поиска сопутствующих товаров методом kNN позволяет выявлять взаимосвязи между блюдами на основе совместных заказов и формировать cross-sale рекомендации. Вместе с тем качество рекомендаций данной модели напрямую зависит от объёма накопленной статистики заказов и плотности матрицы взаимодействий.

SVD-модель обеспечивает формирование персональных рекомендаций на основе истории взаимодействий пользователей с объектами меню и скрытых факторов пользовательских предпочтений. При этом collaborative filtering модели испытывают сложности при работе с новыми пользователями и объектами, для которых отсутствует история взаимодействий.

В связи с этим в рамках работы был реализован гибридный подход, объединяющий несколько моделей рекомендаций в единую систему. Использование гибридной архитектуры позволяет компенсировать ограничения отдельных методов и повысить качество формируемых рекомендаций.

В разработанной системе выбор модели рекомендаций осуществляется в зависимости от сценария использования. Content-based модель применяется для поиска похожих блюд и обработки сценариев cold start. Модель kNN используется для формирования рекомендаций сопутствующих товаров и реализации cross-sale сценариев. SVD-модель применяется для построения персональных рекомендаций на основе пользовательских предпочтений.

Для реализации логики выбора модели использовался механизм определения типа рекомендательного сценария. В зависимости от наличия пользовательской истории взаимодействий и типа запроса система перенаправляет обработку к соответствующей модели рекомендаций.

Ниже приведён фрагмент программного кода:

```
if new_user:
    use_content_based()
```

```
elif need_cross_sale:
    use_knn()

else:
    use_svd()
```

Использование гибридного подхода позволяет объединить преимущества различных методов рекомендаций и обеспечить более гибкое формирование рекомендаций в рамках интеллектуальной системы поддержки продаж кафе.

3.5 Используемые инструменты разработки

Для реализации и исследования рекомендательной системы использовались современные программные средства и библиотеки машинного обучения, предназначенные для обработки данных, анализа пользовательского поведения и построения рекомендательных моделей.

Основным языком программирования, используемым в рамках исследования, выступал язык Python. Выбор Python обусловлен широким распространением данного языка в области анализа данных и машинного обучения, а также наличием большого количества специализированных библиотек для работы с рекомендательными системами.

Разработка и выполнение программного кода осуществлялись в облачной среде Google Colab. Использование Google Colab позволило выполнять обработку данных и исследование моделей без необходимости настройки локального программного окружения. Дополнительным преимуществом среды является поддержка Jupyter Notebook, обеспечивающая удобную работу с кодом, визуализацией данных и промежуточными результатами вычислений.

В процессе реализации рекомендательной системы использовались следующие программные библиотеки:

- pandas – для загрузки, обработки и анализа наборов данных,
- numpy – для выполнения математических операций и работы с массивами данных,

- `scikit-learn` – для реализации методов машинного обучения и обработки текстовых признаков,

- `matplotlib` – для построения графиков и визуализации результатов анализа данных.

Для хранения и обработки данных использовались CSV-файлы. Данный формат был выбран благодаря простоте структуры, удобству загрузки в программную среду и совместимости с используемыми библиотеками анализа данных.

В рамках исследования подготовленные наборы данных использовались для реализации следующих типов рекомендательных моделей:

- коллаборативной фильтрации на основе метода SVD,
- content-based модели,
- модели рекомендаций сопутствующих товаров на основе метода k-ближайших соседей.

Полученные модели в дальнейшем могут быть интегрированы в сервер рекомендаций, взаимодействующий с основным Java-приложением информационной системы кафе. Взаимодействие между системой рекомендаций и клиентским приложением может осуществляться посредством передачи пользовательских параметров и получения сформированных рекомендаций.

Таким образом, можно сделать вывод, что выбранные инструменты и программная среда обеспечивают возможность эффективной обработки данных, исследования пользовательского поведения и реализации рекомендательных моделей в рамках разрабатываемой информационной системы.

4. Экспериментальное исследование моделей рекомендаций

4.1 Цель и постановка эксперимента

После реализации моделей рекомендаций необходимо провести экспериментальное исследование качества их работы в рамках рекомендательной системы заведений общественного питания. Экспериментальная часть исследования направлена на оценку эффективности различных подходов к формированию рекомендаций, а также анализ особенностей их применения в различных пользовательских сценариях.

Целью экспериментального исследования является оценка качества моделей рекомендаций, реализованных в рамках интеллектуальной рекомендательной системы заведений общественного питания, а также анализ их применимости для решения различных задач формирования рекомендаций.

В рамках исследования рассматриваются следующие модели рекомендаций:

- content-based модель, предназначенная для формирования рекомендаций похожих объектов меню на основе характеристик блюд,
- модель поиска сопутствующих товаров методом kNN, используемая для формирования cross-sale рекомендаций на основе совместных заказов,
- коллаборативная фильтрация на основе SVD, предназначенная для построения персональных рекомендаций на основе пользовательских предпочтений и истории взаимодействий.

Основными задачами экспериментального исследования являются:

- оценка качества рекомендаций, формируемых различными моделями,
- анализ преимуществ и ограничений отдельных подходов к формированию рекомендаций,
- исследование работы моделей в различных пользовательских сценариях,
- анализ проблемы холодного старта,

– исследование влияния объёма пользовательских взаимодействий на качество персональных предложений, сгенерированных с помощью модели SVD,

– анализ эффективности гибридного подхода к формированию рекомендаций.

Для оценки качества работы рекомендательных моделей используются как количественные, так и качественные методы анализа. Для модели на основе SVD применяются метрики RMSE и MAE, позволяющие оценить точность предсказания пользовательских рейтингов. Для content-based модели и модели поиска сопутствующих товаров kNN проводится качественный анализ релевантности и логической согласованности формируемых рекомендаций.

В рамках главы будут проведены эксперименты, направленные на сравнение моделей рекомендаций, исследование проблемы холодного старта, оценку качества рекомендаций сопутствующих товаров, а также анализ влияния объёма данных на качество работы SVD-модели в рамках интеллектуальной рекомендательной системы заведений общественного питания.

4.2 Описание набора данных

В рамках исследования использовались несколько наборов данных, подготовленных на этапе предварительной обработки информации. Данные наборы применялись для обучения, тестирования и анализа качества моделей рекомендаций, реализованных в рамках рекомендательной системы заведений общественного питания.

Для построения контентно-ориентированной модели рекомендаций использовался набор данных `menu_dataset`, содержащий информацию о характеристиках объектов меню. В состав набора входили наименования блюд, категории, ингредиенты, а также дополнительные признаки, использовавшиеся для формирования векторного представления объектов и расчёта степени сходства между блюдами.

Для реализации модели рекомендаций сопутствующих товаров методом kNN использовался набор данных `orders_dataset`, содержащий информацию о пользовательских заказах и совместных покупках блюд. Данный набор данных применялся для выявления взаимосвязей между объектами меню и формирования рекомендаций сопутствующих товаров.

Для обучения модели коллаборативной фильтрации SVD использовался набор данных `ratings_dataset`, содержащий информацию о взаимодействиях пользователей с объектами меню. Набор включал идентификаторы пользователей, идентификаторы блюд и соответствующие рейтинги взаимодействий, используемые для построения персональных рекомендаций (таблица 12).

Таблица 12 – Характеристика экспериментальных данных

Набор данных	Назначение	Основные данные
<code>menu_dataset</code>	Контентно-ориентированная модель	Характеристики и признаки блюд
<code>orders_dataset</code>	Модель рекомендаций сопутствующих товаров kNN	Пользовательские заказы и совместные покупки
<code>ratings_dataset</code>	Модель коллаборативной фильтрации SVD	Пользовательские рейтинги взаимодействий

Дополнительно в рамках исследования использовались количественные характеристики экспериментальных данных, представленные в таблице 13.

Таблица 13 – Количественные характеристики данных

Параметр	Значение
Количество пользователей	100
Количество блюд	100
Количество заказов	798
Диапазон рейтингов	1-5

Используемые наборы данных характеризуются ограниченным объёмом пользовательских взаимодействий, что соответствует условиям функционирования реальных заведений общественного питания малого и среднего масштаба. Ограниченность данных оказывает влияние на качество моделей рекомендаций, особенно моделей коллаборативной фильтрации, чувствительных к плотности пользовательских взаимодействий.

Подготовленные наборы данных использовались для проведения экспериментального исследования моделей рекомендаций, анализа качества формируемых рекомендаций и оценки применимости различных подходов к построению интеллектуальной рекомендательной системы заведений общественного питания.

4.3 Методы оценки качества

Для оценки качества работы моделей рекомендаций использовались количественные и качественные методы анализа. Выбор методов оценки осуществлялся с учётом особенностей реализованных рекомендательных моделей и принципов формирования рекомендаций.

Для оценки качества модели коллаборативной фильтрации на основе SVD использовались количественные метрики RMSE и MAE, позволяющие определить точность предсказания пользовательских рейтингов [12].

Метрика RMSE (Root Mean Squared Error – среднеквадратичная ошибка) характеризует среднее отклонение предсказанных рейтингов от реальных значений пользовательских оценок [12]. Данная метрика более чувствительна к большим ошибкам предсказания, поскольку ошибки предварительно возводятся в квадрат.

Формула расчёта RMSE представлена ниже.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}, \quad (5)$$

где y_i – реальное значение рейтинга;

$\hat{y}_i$ – предсказанное значение рейтинга;

n – количество предсказаний.

Чем меньше значение RMSE, тем выше точность модели коллаборативной фильтрации при прогнозировании пользовательских предпочтений.

Для дополнительной оценки качества рекомендаций использовалась метрика MAE (Mean Absolute Error – средняя абсолютная ошибка),

характеризующая среднее абсолютное отклонение предсказанных рейтингов от реальных значений [12].

Формула расчёта MAE представлена ниже.

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}. \quad (6)$$

где y_i - реальное значение рейтинга;

$\hat{y}_i$ - предсказанное значение рейтинга;

n - количество предсказаний.

Метрика MAE позволяет определить среднюю величину ошибки предсказания без дополнительного усиления крупных отклонений, что делает её удобной для интерпретации результатов работы рекомендательной модели.

Для оценки качества контентно-ориентированной модели и модели рекомендаций сопутствующих товаров использовались качественные методы анализа. Поскольку данные модели формируют рекомендации на основе сходства объектов и совместных пользовательских взаимодействий, применение метрик предсказания рейтингов для них является ограниченным.

В рамках качественного анализа оценивались:

- релевантность формируемых рекомендаций,
- логическая согласованность рекомендуемых объектов,
- соответствие рекомендаций пользовательскому сценарию,
- интерпретируемость результатов рекомендаций.

При анализе рекомендаций сопутствующих товаров оценивалась корректность формирования cross-sale рекомендаций на основе совместных заказов пользователей. Для контентно-ориентированной модели анализировалась степень сходства рекомендуемых объектов меню по их характеристикам и признакам.

Использование количественных и качественных методов оценки позволяет комплексно исследовать эффективность различных моделей рекомендаций в рамках интеллектуальной рекомендательной системы заведений общественного питания.

4.4 Результаты экспериментального исследования

4.4.1 Сравнительный анализ моделей рекомендаций

Целью данного исследования является сравнительный анализ реализованных моделей рекомендаций в рамках рекомендательной системы заведений общественного питания, а также определение особенностей их применения в различных пользовательских сценариях.

В рамках исследования анализировались три модели рекомендаций:

- контентно-ориентированная модель, формирующая рекомендации на основе характеристик и признаков объектов меню,
- модель рекомендаций сопутствующих товаров методом kNN, использующая информацию о совместных заказах пользователей,
- модель коллаборативной фильтрации на основе SVD, формирующая персональные рекомендации на основе пользовательских предпочтений и истории взаимодействий.

Сравнение моделей выполнялось по нескольким критериям, отражающим особенности работы рекомендательных алгоритмов в условиях заведений общественного питания (таблица 14).

Таблица 14 – Сравнение моделей рекомендаций

Критерий	Контентно-ориентированная модель	kNN	SVD
Персонализация	Низкая	Средняя	Высокая
Работа при холодном старте	Высокая	Низкая	Низкая
Формирование рекомендаций сопутствующих товаров	Низкая	Высокая	Средняя
Требования к объёму данных	Низкие	Средние	Высокие
Интерпретируемость рекомендаций	Высокая	Высокая	Средняя

Контентно-ориентированная модель продемонстрировала высокую эффективность при формировании рекомендаций похожих объектов меню. Основным преимуществом данного подхода является возможность работы в условиях отсутствия пользовательской истории взаимодействий, что делает модель эффективной для решения проблемы холодного старта. Вместе с тем

рекомендации формируются исключительно на основе характеристик объектов меню, без учёта индивидуальных пользовательских предпочтений.

Модель рекомендаций сопутствующих товаров методом kNN показала высокую эффективность при формировании cross-sale рекомендаций. Использование информации о совместных заказах позволяет выявлять взаимосвязи между объектами меню и предлагать пользователю логически связанные товары. Качество рекомендаций данной модели напрямую зависит от объёма накопленной статистики заказов и плотности пользовательских взаимодействий.

Модель коллаборативной фильтрации на основе SVD обеспечивает наиболее высокий уровень персонализации рекомендаций. Использование пользовательских рейтингов и скрытых факторов предпочтений позволяет формировать рекомендации с учётом особенностей поведения пользователей. При этом качество работы модели существенно зависит от объёма доступных пользовательских взаимодействий и наличия истории заказов.

В ходе исследования были получены примеры рекомендаций, сформированных различными моделями (таблица 15).

Таблица 15 – Примеры рекомендаций различных моделей

Модель	Исходные данные	Пример рекомендаций
Контентно-ориентированная	Капучино	Латте, Раф, Эспрессо
kNN	Капучино	Чизкейк, Круассан
SVD	Пользователь №84	Раф, Лимонад

Полученные результаты демонстрируют, что различные модели рекомендаций эффективны для различных пользовательских сценариев. Контентно-ориентированная модель лучше подходит для формирования рекомендаций похожих объектов меню и решения проблемы холодного старта. Модель kNN наиболее эффективно применяется для формирования рекомендаций сопутствующих товаров. Модель коллаборативной фильтрации SVD обеспечивает наиболее высокий уровень персонализации рекомендаций.

Проведённое исследование подтверждает целесообразность использования гибридного подхода в рамках интеллектуальной

рекомендательной системы заведений общественного питания, поскольку объединение различных моделей позволяет компенсировать ограничения отдельных методов формирования рекомендаций.

4.4.2 Исследование проблемы холодного старта

Одной из основных проблем рекомендательных систем является проблема холодного старта, возникающая при отсутствии или недостаточности пользовательской истории взаимодействий. В условиях заведений общественного питания данная проблема является особенно актуальной, поскольку новые пользователи могут не иметь ранее сформированных заказов или пользовательских оценок.

Целью данного исследования является анализ эффективности различных моделей рекомендаций при отсутствии или ограниченном объёме пользовательских взаимодействий.

В рамках исследования рассматривались два сценария холодного старта:

- новый пользователь, полностью отсутствующий в истории взаимодействий,
- пользователь с ограниченным количеством заказов.

Для моделирования полного холодного старта использовался условный пользователь, отсутствующий в наборе пользовательских рейтингов `ratings_dataset`. В ходе эксперимента была выполнена попытка формирования рекомендаций с использованием модели коллаборативной фильтрации SVD.

Результаты исследования показали, что при отсутствии пользовательской истории модель SVD не способна формировать полноценные персональные рекомендации. Несмотря на возможность получения прогнозируемых рейтингов, формируемые значения отражают усреднённые закономерности обучающего набора данных и не учитывают индивидуальные предпочтения пользователя (рисунок 25).

... SVD-рекомендации для нового пользователя 125:							
	user_id	item_id	predicted_rating	name	category	price	
0	125	50	1.933	Самса с курицей	Выпечка	160	
1	125	76	1.901	Компот из сухофруктов	Напиток	80	
2	125	81	1.897	Лимонад	Напиток	110	
3	125	80	1.877	Минеральная вода	Напиток	80	
4	125	87	1.834	Горчица	Добавка	25	

Рисунок 25 – Результаты работы SVD-модели для нового пользователя без истории взаимодействий

Для сравнения была использована контентно-ориентированная модель рекомендаций, формирующая рекомендации на основе характеристик объектов меню. В качестве исходного объекта использовалось блюдо «Капучино», на основании которого система формировала рекомендации похожих объектов меню.

В результате работы контентно-ориентированной модели были получены рекомендации схожих напитков, включая «Латте», «Раф» и «Эспрессо». Полученные рекомендации характеризуются высокой логической согласованностью и не требуют наличия пользовательской истории взаимодействий (рисунок 26).

Content-based рекомендации для нового пользователя 125 на основе блюда: Капучино							
	item_id	name	category	cuisine	price	similarity	
63	64	Латте	Кофе	Европейская	170	1.000000	
67	68	Кофе с сиропом	Кофе	Европейская	180	0.880999	
66	67	Айс-латте	Кофе	Европейская	180	0.868350	
64	65	Раф	Кофе	Европейская	190	0.788455	
60	61	Эспрессо	Кофе	Европейская	100	0.774258	

Рисунок 26 – Результаты работы контентно-ориентированной модели при холодном старте

Дополнительно был рассмотрен сценарий частичного холодного старта, при котором пользователь обладает ограниченным количеством взаимодействий с системой. Для проведения исследования случайным образом выбирался пользователь с минимальной историей заказов.

В ходе эксперимента было установлено, что при ограниченном объёме пользовательских взаимодействий качество персональных рекомендаций SVD-модели остаётся нестабильным, поскольку модели недостаточно информации для точного выявления пользовательских предпочтений.

Контентно-ориентированная модель в данном сценарии продолжала формировать релевантные рекомендации на основе характеристик ранее выбранных объектов меню (таблица 16).

Таблица 16 – Сравнение моделей при холодном старте

Модель	Новый пользователь	Пользователь с малой историей
Контентно-ориентированная модель	Высокая эффективность	Высокая эффективность
kNN	Низкая эффективность	Средняя эффективность
SVD	Низкая эффективность	Средняя эффективность

Полученные результаты (таблица 16) показывают, что контентно-ориентированный подход является наиболее устойчивым к проблеме холодного старта, поскольку формирование рекомендаций выполняется на основе характеристик объектов меню, а не пользовательской истории взаимодействий.

Проведённое исследование также подтверждает целесообразность применения гибридного подхода в рамках рекомендательной системы заведений общественного питания, при котором для новых пользователей могут использоваться контентно-ориентированные рекомендации, а персональные методы коллаборативной фильтрации применяются по мере накопления пользовательской истории взаимодействий.

4.4.3 Анализ рекомендаций сопутствующих товаров

В рамках интеллектуальной рекомендательной системы заведений общественного питания рекомендации сопутствующих товаров представляют практическую ценность, поскольку позволяют формировать дополнительные продажи, повышать средний чек заказа и улучшать пользовательский опыт при взаимодействии с системой оформления заказов.

В отличие от персональных рекомендаций, ориентированных на долгосрочные пользовательские предпочтения, рекомендации сопутствующих товаров формируются непосредственно в процессе оформления заказа и направлены на предложение логически связанных объектов меню.

Целью данного исследования является анализ качества рекомендаций сопутствующих товаров, формируемых моделью kNN на основе совместных пользовательских заказов.

Для проведения исследования использовался набор данных `orders_dataset`, содержащий информацию о пользовательских заказах и совместных покупках объектов меню. Формирование рекомендаций выполнялось на основе анализа частоты совместного появления блюд в пользовательских заказах.

Принцип работы модели основан на предположении о том, что объекты меню, часто встречающиеся в одних и тех же заказах, обладают логической взаимосвязью и могут использоваться в качестве сопутствующих рекомендаций. Например, при заказе напитков система может рекомендовать десерты или выпечку, а при заказе основных блюд – салаты или дополнительные позиции меню (рисунок 27).

В ходе исследования были сформированы рекомендации сопутствующих товаров для различных объектов меню (таблица 17).

Таблица 17 – Примеры рекомендаций сопутствующих товаров

Исходный объект меню	Рекомендации
Капучино	Чизкейк, Круассан, Раф
Чай	Медовик, Булочка
Борщ	Салат, Хлеб
Латте	Тирамису, Печенье

В результате исследования было установлено, что модель kNN позволяет формировать логически согласованные рекомендации на основе пользовательских заказов. Полученные рекомендации соответствуют типичным сценариям поведения пользователей в заведениях общественного питания и отражают реальные взаимосвязи между объектами меню.

... Результаты тестирования kNN-модели:

	source_item	recommended_item	similarity
0	Сок яблочный	Борщ	0.197
1	Сок яблочный	Овощное рагу	0.179
2	Сок яблочный	Рассольник	0.161
3	Сок яблочный	Макароны с сыром	0.157
4	Сок яблочный	Сельдь под шубой	0.144
5	Морковь по-корейски	Гречка отварная	0.177
6	Морковь по-корейски	Тефтели с рисом	0.131
7	Морковь по-корейски	Сок яблочный	0.129

Рисунок 27 – Пример формирования рекомендаций сопутствующих товаров методом kNN

Особую практическую значимость рекомендации сопутствующих товаров представляют для систем электронного меню и онлайн-заказов, поскольку позволяют предлагать дополнительные позиции непосредственно в процессе оформления заказа.

В ходе анализа также было установлено, что качество рекомендаций зависит от объёма накопленной статистики пользовательских заказов. Для редко заказываемых объектов меню количество доступных взаимосвязей оказывается ограниченным, что снижает качество формируемых рекомендаций.

Полученные результаты показывают, что модель рекомендаций сопутствующих товаров методом kNN эффективно решает задачу формирования cross-sale рекомендаций в рамках интеллектуальной рекомендательной системы заведений общественного питания и может использоваться в качестве одного из компонентов гибридного рекомендательного подхода.

4.4.4 Исследование влияния количества данных для SVD

Качество моделей коллаборативной фильтрации напрямую зависит от объёма пользовательских взаимодействий, используемых в процессе обучения. Для формирования персональных рекомендаций модели необходимо выявлять скрытые закономерности пользовательского поведения,

что требует достаточного количества данных о взаимодействиях пользователей с объектами меню [9].

Целью данного исследования является анализ влияния объёма пользовательских взаимодействий на качество персональных рекомендаций модели коллаборативной фильтрации SVD.

В рамках исследования использовался набор пользовательских рейтингов `ratings_dataset`, содержащий информацию о взаимодействиях пользователей с объектами меню. Для проведения эксперимента были сформированы выборки различного объёма, содержащие 30%, 60% и 100% пользовательских взаимодействий.

Для обеспечения воспроизводимости результатов при формировании выборок использовалось фиксированное значение параметра `random_state`.

Для каждой выборки выполнялось обучение модели SVD и последующая оценка качества рекомендаций с использованием метрик RMSE и MAE (таблица 18).

Таблица 18 – Результаты исследования влияния объёма данных на качество SVD-модели

Объём данных	Количество записей	RMSE	MAE
30%	455	0,933	0,680
60%	1063	0,887	0,681
100%	1639	0,848	0,626

Полученные результаты демонстрируют зависимость качества модели коллаборативной фильтрации от объёма пользовательских взаимодействий [9]. При увеличении количества данных наблюдается снижение значения RMSE, что свидетельствует о повышении точности предсказания пользовательских рейтингов.

Наилучшие результаты были получены при использовании полного набора данных, содержащего 100% пользовательских взаимодействий. В данном случае модель продемонстрировала минимальные значения RMSE и MAE, что указывает на более точное выявление пользовательских предпочтений и улучшение качества персональных рекомендаций.

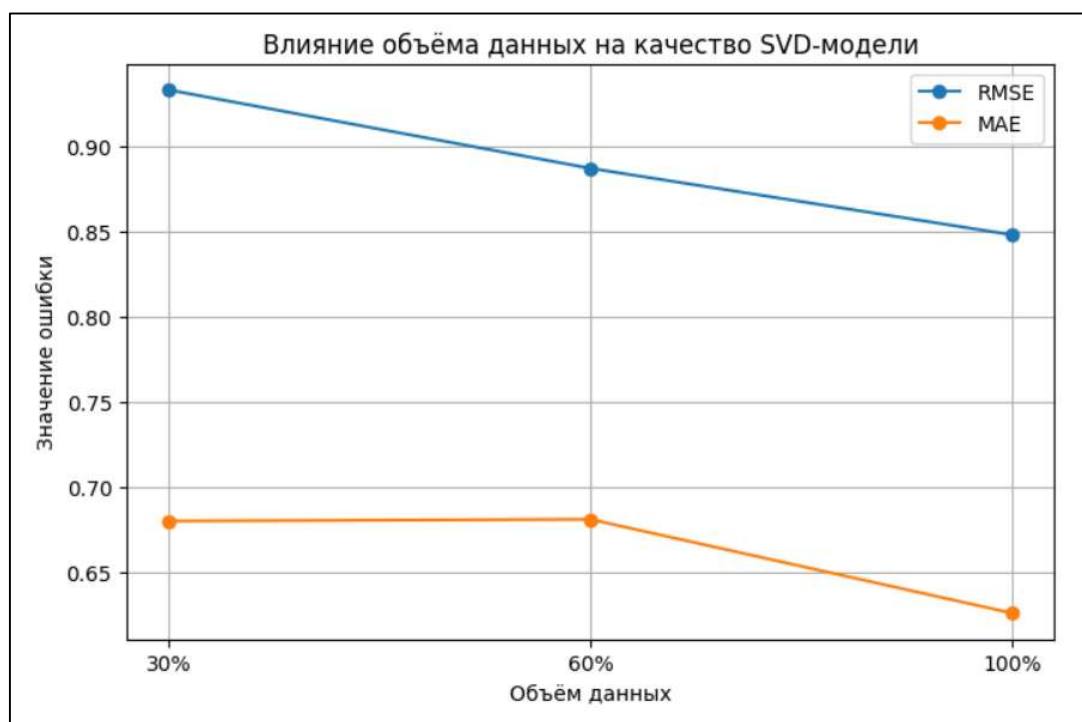


Рисунок 28 – Изменение значений RMSE и MAE при увеличении объёма данных

В ходе исследования также было установлено, что различия между значениями метрики MAE для частичных выборок выражены слабее, чем различия RMSE. Это может быть связано с ограниченным объёмом экспериментального набора данных и разреженным характером пользовательских взаимодействий.

Полученные результаты подтверждают, что эффективность модели коллаборативной фильтрации существенно зависит от объёма пользовательских данных. При недостаточном количестве взаимодействий качество персональных рекомендаций снижается, что ограничивает возможности применения модели в условиях малого объёма пользовательской активности.

Проведённое исследование также подтверждает целесообразность использования гибридного подхода в рамках интеллектуальной рекомендательной системы заведений общественного питания, при котором модели коллаборативной фильтрации могут дополняться контентно-ориентированными методами, менее чувствительными к ограниченному объёму пользовательских взаимодействий.

4.5 Сравнительный анализ

На основании проведённых экспериментальных исследований был выполнен сравнительный анализ реализованных моделей рекомендаций в рамках интеллектуальной рекомендательной системы заведений общественного питания.

Проведённые исследования показали, что различные подходы к формированию рекомендаций обладают собственными преимуществами и ограничениями, а также ориентированы на решение различных пользовательских сценариев.

Контентно-ориентированная модель продемонстрировала высокую эффективность при формировании рекомендаций похожих объектов меню и решении проблемы холодного старта. Основным преимуществом данного подхода является возможность формирования рекомендаций без необходимости наличия пользовательской истории взаимодействий. Вместе с тем модель обладает ограниченными возможностями персонализации, поскольку рекомендации формируются исключительно на основе характеристик объектов меню.

Модель рекомендаций сопутствующих товаров методом kNN показала высокую эффективность при анализе совместных пользовательских заказов и формировании cross-sale рекомендаций. Использование информации о совместных покупках позволяет предлагать пользователю логически связанные объекты меню непосредственно в процессе оформления заказа. Данный подход обладает высокой практической значимостью для заведений общественного питания, поскольку способствует формированию дополнительных продаж и повышению среднего чека заказа.

Модель коллаборативной фильтрации на основе SVD продемонстрировала наиболее высокий уровень персонализации рекомендаций. Использование пользовательских рейтингов и скрытых факторов предпочтений позволяет учитывать особенности пользовательского поведения и формировать индивидуальные рекомендации. Вместе с тем

проведённое исследование показало высокую зависимость качества модели от объёма пользовательских взаимодействий и наличие ограничений при решении проблемы холодного старта.

Таблица 20 – Области эффективного применения моделей рекомендаций

Модель	Основная задача	Преимущества	Ограничения
Content-based	Рекомендации похожих объектов меню	Устойчивость к проблеме холодного старта, высокая интерпретируемость	Ограниченная персонализация
kNN	Рекомендации сопутствующих товаров	Эффективность cross-sale рекомендаций, поддержка допродаж	Зависимость от статистики совместных заказов
SVD	Персональные рекомендации	Высокая точность персонализации	Высокая зависимость от объёма пользовательских данных

Результаты исследования показывают, что универсальной модели рекомендаций, одинаково эффективно решающей все задачи интеллектуальной рекомендательной системы заведений общественного питания, не существует. Каждая из реализованных моделей обладает собственной областью эффективного применения и ориентирована на решение отдельных задач формирования рекомендаций.

Контентно-ориентированная модель наиболее эффективно применяется при отсутствии пользовательской истории взаимодействий и формировании рекомендаций похожих объектов меню. Модель kNN целесообразно использовать для формирования рекомендаций сопутствующих товаров в процессе оформления заказа. Модель коллаборативной фильтрации SVD наиболее эффективна для построения персональных рекомендаций при наличии достаточного объёма пользовательских взаимодействий.

Полученные результаты подтверждают целесообразность использования гибридного подхода в рамках интеллектуальной рекомендательной системы заведений общественного питания [1], при котором различные модели рекомендаций могут использоваться совместно для решения различных

пользовательских сценариев и компенсации ограничений отдельных методов формирования рекомендаций.

4.6 Оценка эффективности гибридного подхода

Проведённые экспериментальные исследования показали, что различные модели рекомендаций обладают собственными преимуществами и ограничениями, а также ориентированы на решение различных пользовательских сценариев в рамках интеллектуальной рекомендательной системы заведений общественного питания.

Контентно-ориентированная модель эффективно решает задачу формирования рекомендаций похожих объектов меню и демонстрирует устойчивость к проблеме холодного старта. Модель рекомендаций сопутствующих товаров методом kNN позволяет выявлять взаимосвязи между объектами меню и формировать рекомендации сопутствующих товаров в процессе оформления заказа. Модель коллаборативной фильтрации SVD обеспечивает высокий уровень персонализации рекомендаций при наличии достаточного объёма пользовательских взаимодействий.

В ходе исследования было установлено, что использование только одной рекомендательной модели не позволяет эффективно решать все задачи формирования рекомендаций одновременно. Ограничения отдельных моделей оказывают влияние на качество рекомендаций в различных пользовательских сценариях.

Использование гибридного подхода позволяет объединить преимущества различных рекомендательных моделей и компенсировать ограничения отдельных методов формирования рекомендаций [13].

В рамках интеллектуальной рекомендательной системы заведений общественного питания контентно-ориентированная модель может использоваться при отсутствии пользовательской истории взаимодействий, модель kNN – для формирования рекомендаций сопутствующих товаров в процессе оформления заказа, а модель SVD – для построения

персонализированных рекомендаций на основе пользовательских предпочтений пользователей.

Таким образом, применение гибридного подхода позволяет повысить гибкость рекомендательной системы, обеспечить поддержку различных пользовательских сценариев и улучшить качество формируемых рекомендаций. Полученные результаты подтверждают целесообразность совместного использования различных моделей рекомендаций при разработке рекомендательных систем заведений общественного питания.

ЗАКЛЮЧЕНИЕ

В рамках магистерской диссертации было проведено исследование методов построения интеллектуальной рекомендательной системы заведений общественного питания. Актуальность темы обусловлена необходимостью повышения качества пользовательского взаимодействия с цифровыми сервисами заведений общественного питания, а также потребностью в автоматизации процессов формирования персонализированных рекомендаций объектов меню.

Целью работы являлось исследование и разработка интеллектуальной рекомендательной системы заведений общественного питания на основе различных подходов к формированию рекомендаций.

В ходе выполнения работы был проведён анализ предметной области и рассмотрены современные подходы к построению рекомендательных систем. Были исследованы особенности контентно-ориентированных методов, методов коллаборативной фильтрации, а также подходов, основанных на анализе совместных пользовательских взаимодействий.

В рамках практической части исследования были подготовлены экспериментальные наборы данных, содержащие информацию об объектах меню, пользовательских заказах и пользовательских рейтингах. На основе подготовленных данных были реализованы и исследованы три модели рекомендаций: контентно-ориентированная модель, модель рекомендаций сопутствующих товаров методом kNN и модель коллаборативной фильтрации на основе SVD.

В ходе исследования контентно-ориентированной модели было установлено, что данный подход эффективно решает задачу формирования рекомендаций похожих объектов меню и демонстрирует устойчивость к проблеме холодного старта. Использование характеристик и признаков объектов меню позволяет формировать рекомендации даже при отсутствии пользовательской истории взаимодействий.

Исследование модели рекомендаций сопутствующих товаров методом kNN показало эффективность применения подхода для формирования cross-sale рекомендаций на основе совместных пользовательских заказов. Полученные результаты подтверждают возможность использования модели для повышения среднего чека заказа и формирования дополнительных продаж в заведениях общественного питания.

В рамках исследования модели коллаборативной фильтрации SVD были реализованы механизмы формирования персонализированных рекомендаций на основе пользовательских рейтингов и скрытых факторов предпочтений. Проведённые эксперименты показали, что качество модели существенно зависит от объёма пользовательских взаимодействий. При увеличении количества доступных данных наблюдалось снижение значений ошибок RMSE и MAE, что свидетельствует о повышении точности персонализированных рекомендаций.

Проведённое экспериментальное исследование показало, что различные модели рекомендаций обладают собственными преимуществами и ограничениями. Контентно-ориентированная модель наиболее эффективна при решении проблемы холодного старта, модель kNN – при формировании рекомендаций сопутствующих товаров, а модель SVD – при построении персонализированных рекомендаций на основе пользовательских предпочтений.

Полученные результаты подтверждают, что универсальной модели рекомендаций, одинаково эффективно решающей все задачи интеллектуальной рекомендательной системы заведений общественного питания, не существует. Наиболее эффективным подходом является использование гибридной архитектуры, объединяющей преимущества различных методов формирования рекомендаций и компенсирующей ограничения отдельных моделей.

Практическая значимость работы заключается в возможности использования полученных результатов при разработке интеллектуальных

систем рекомендаций для заведений общественного питания, систем электронного меню и сервисов онлайн-заказов.

Перспективы дальнейшего развития исследования связаны с расширением объёма пользовательских данных, интеграцией рекомендательной системы с мобильными приложениями и веб-сервисами, а также исследованием современных нейросетевых подходов к построению рекомендательных систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фалк, К. Построение систем рекомендаций / К. Фалк; перевод с английского А. А. Слинкина. – Москва: ДМК Пресс, 2020. – 484 с.
2. Баник, Р. Знакомство с системами рекомендаций на основе Python / Р. Баник; перевод с английского А. А. Слинкина. – Москва: ДМК Пресс, 2019. – 326 с.
3. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах; перевод с английского А. А. Слинкина. – Москва: ДМК Пресс, 2015. – 400 с.
4. Брюхов, Б. А. Алгоритмические основы рекомендательных систем: учебно-методическое пособие / Б. А. Брюхов, Е. В. Попова. – Москва: МИРЭА – Российский технологический университет, 2021. – 94 с.
5. Королева, М. Ю. Особенности применения рекомендательных алгоритмов в ресторанном бизнесе / М. Ю. Королева, Д. В. Козлов // Вестник компьютерных и информационных технологий. – 2022. – № 5. – С. 14–22.
6. Система автоматизации r_keeper: обзор программного продукта // SoWare.ru: каталог программного обеспечения. – URL: soware.ru (дата обращения: 22.05.2026).
7. Рекомендованные блюда в r_keeper Kiosk: пользовательская документация r_keeper // r_keeper Docs: База знаний. – URL: rkeeper.ru (дата обращения: 22.05.2026).
8. Цепочки рекомендаций в r_keeper Kiosk: пользовательская документация r_keeper // r_keeper Docs: База знаний. – URL: rkeeper.ru (дата обращения: 22.05.2026).
9. Koren, Y. Matrix Factorization Techniques for Recommender Systems / Y. Koren, R. Bell, C. Volinsky // Computer. – 2009. – Vol. 42, no. 8. – P. 30–37. – URL: wordpress.com (дата обращения: 22.05.2026).
10. Scikit-learn: Machine Learning in Python: сайт. – URL: scikit-learn.org (дата обращения: 22.05.2026).

11. Surprise: A Python library for recommender systems: сайт. – URL: surpriselib.com (дата обращения: 22.05.2026).
12. Ricci, F. Recommender Systems Handbook / F. Ricci, L. Rokach, B. Shapira. – New York: Springer, 2015. – 1003 p.
13. Burke, R. Hybrid Web Recommender Systems / R. Burke // The Adaptive Web. – 2007. – P. 377–408.

ПРИЛОЖЕНИЕ А

(Обязательное)

Рекомендательная система автоматизации работы заведений

общественного питания

Листинг программы

Файл recommendation_system.py

1. Загрузка датасетов

```
"""
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
menu_df = pd.read_csv("sample_data/menu_dataset_cafe_100.csv")
orders_df =
pd.read_csv("sample_data/orders_dataset_cafe_v2.csv")
ratings_df =
pd.read_csv("sample_data/ratings_dataset_cafe_v2.csv")
```

```
print("Размер menu_dataset:", menu_df.shape)
print("Размер orders_dataset:", orders_df.shape)
print("Размер ratings_dataset:", ratings_df.shape)
```

```
display(menu_df.head())
display(orders_df.head())
display(ratings_df.head())
```

"""2. Проверка пропусков"""

```
print("Пропуски в menu_dataset:")
print(menu_df.isnull().sum())
```

```
print("\nПропуски в orders_dataset:")
print(orders_df.isnull().sum())
```

```
print("\nПропуски в ratings_dataset:")
print(ratings_df.isnull().sum())
```

"""3. Проверка дубликатов"""

```
# Очистка дубликатов, если они есть
menu_df = menu_df.drop_duplicates()
orders_df = orders_df.drop_duplicates()
ratings_df = ratings_df.drop_duplicates()
```

```
print("Дубликаты в menu_dataset:", menu_df.duplicated().sum())
print("Дубликаты в orders_dataset:",
orders_df.duplicated().sum())
print("Дубликаты в ratings_dataset:",
ratings_df.duplicated().sum())
```

"""4. Проверка типов данных"""

```
print("Типы данных menu_dataset:")
print(menu_df.dtypes)
```

```

print("\nТипы данных orders_dataset:")
print(orders_df.dtypes)

print("\nТипы данных ratings_dataset:")
print(ratings_df.dtypes)

"""5. Проверка наличия некорректных значений"""

# Проверка отрицательных или нулевых цен
invalid_prices = menu_df[menu_df["price"] <= 0]

if invalid_prices.empty:
    print("Некорректных значений цены не обнаружено.")
else:
    print("Обнаружены некорректные значения цены:")
    display(invalid_prices)

# Проверка quantity <= 0
invalid_quantity = orders_df[orders_df["quantity"] <= 0]

if invalid_quantity.empty:
    print("Некорректных значений quantity не обнаружено.")
else:
    print("Обнаружены некорректные значения quantity:")
    display(invalid_quantity)

# Проверка рейтингов вне диапазона 1-5
invalid_ratings = ratings_df[
    (ratings_df["rating"] < 1) |
    (ratings_df["rating"] > 5)
]

if invalid_ratings.empty:
    print("Некорректных значений rating не обнаружено.")
else:
    print("Обнаружены некорректные значения rating:")
    display(invalid_ratings)

"""6. Формирование признака features для Content-based"""

# Проверяем наличие необходимых столбцов
required_columns = ["category", "cuisine", "ingredients",
                    "tags"]

missing_columns = [
    column for column in required_columns
    if column not in menu_df.columns
]

if missing_columns:
    print("Невозможно сформировать признак features.")

```

```

        print("Отсутствуют столбцы:", missing_columns)
    else:
        menu_df["features"] = (
            menu_df["category"].astype(str) + " " +
            menu_df["cuisine"].astype(str) + " " +
            menu_df["ingredients"].astype(str) + " " +
            menu_df["tags"].astype(str)
        )

        print("Признак features успешно сформирован.")
        print("Количество записей:", len(menu_df))
        print("\nПример сформированных признаков:")

        display(menu_df[["item_id", "name", "features"]].head(10))

"""7. TF-IDF-векторизация признаков блюд"""

from sklearn.feature_extraction.text import TfidfVectorizer

if "features" not in menu_df.columns:
    print("Столбец features не найден. Сначала необходимо сформировать признаки блюд.")
else:
    tfidf = TfidfVectorizer()
    tfidf_matrix = tfidf.fit_transform(menu_df["features"])

    print("TF-IDF матрица успешно сформирована.")
    print("Размер матрицы:", tfidf_matrix.shape)
    print("Количество блюд:", tfidf_matrix.shape[0])
    print("Количество текстовых признаков:",
          tfidf_matrix.shape[1])

"""8. Проверка разреженности матрицы user-item для SVD"""

num_users = ratings_df["user_id"].nunique()
num_items = ratings_df["item_id"].nunique()

possible_interactions = num_users * num_items
actual_interactions = len(ratings_df)

sparsity = 1 - (actual_interactions / possible_interactions)

print("Анализ матрицы пользователь-объект:")
print("Количество пользователей:", num_users)
print("Количество блюд:", num_items)
print("Возможное количество взаимодействий:",
      possible_interactions)
print("Фактическое количество взаимодействий:",
      actual_interactions)
print(f"Разреженность матрицы: {sparsity:.2%}")

"""9. График распределения рейтингов"""

rating_counts = ratings_df["rating"].value_counts().sort_index()

```

```
plt.figure(figsize=(6, 4))
rating_counts.plot(kind="bar")
plt.title("Распределение рейтингов")
plt.xlabel("Рейтинг")
plt.ylabel("Количество взаимодействий")
plt.grid(axis="y")
plt.xticks(rotation=0)
plt.show()
```

"""10. График популярных категорий"""

```
category_counts = orders_df["category"].value_counts()

plt.figure(figsize=(8, 4))
category_counts.plot(kind="bar")
plt.title("Распределение заказов по категориям")
plt.xlabel("Категория")
plt.ylabel("Количество заказанных позиций")
plt.xticks(rotation=45, ha="right")
plt.grid(axis="y")
plt.show()
```

"""11. Самые популярные блюда"""

```
popular_items = (
    orders_df
    .groupby(["item_id", "item_name"])["quantity"]
    .sum()
    .reset_index()
    .sort_values(by="quantity", ascending=False)
)

display(popular_items.head(10))
```

"""12. Сохранение предобработанного меню"""

```
menu_df.to_csv("menu_dataset_preprocessed.csv", index=False,
encoding="utf-8-sig")
ratings_df.to_csv("ratings_dataset_preprocessed.csv",
index=False, encoding="utf-8-sig")
orders_df.to_csv("orders_dataset_preprocessed.csv", index=False,
encoding="utf-8-sig")
```

"""# **3.3 ФОРМИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ПРЕДПОЧТЕНИЙ**

3.3.1. Анализ пользовательской активности
"""

Анализ пользовательской активности

```
user_activity = (
    orders_df.groupby("user_id")
    .agg(
```

```

        orders_count=("order_id", "nunique"),
        items_count=("quantity", "sum"),
        unique_items=("item_id", "nunique")
    )
    .reset_index()
)

print("Пользовательская активность успешно рассчитана.")
print("Количество пользователей:", len(user_activity))

display(user_activity.head(10))

"""3.3.2. Расчёт предпочтений пользователей по категориям
блюд"""

# Расчёт предпочтений пользователей по категориям блюд

user_category_preferences = (
    orders_df
    .groupby(["user_id", "category"])
    .agg(
        category_quantity=("quantity", "sum"),
        category_orders=("order_id", "nunique")
    )
    .reset_index()
)

# Сортируем: сначала пользователь, затем самые часто
заказываемые категории
user_category_preferences =
user_category_preferences.sort_values(
    by=["user_id", "category_quantity"],
    ascending=[True, False]
)

print("Предпочтения пользователей по категориям успешно
рассчитаны.")
display(user_category_preferences.head(10))

"""3.3.3 Топ-3 любимых категории для каждого пользователя"""

# Топ-3 категории для каждого пользователя

top_categories = (
    user_category_preferences
    .groupby("user_id")
    .head(3)
    .groupby("user_id")["category"]
    .apply(lambda x: ", ".join(x))
    .reset_index()
)

top_categories = top_categories.rename(
    columns={"category": "favorite_categories"}
)

```

```

)

print("Топ-3 категории для каждого пользователя сформированы.")
display(top_categories.head(10))

"""3.3.4 Объединение активности пользователя и любимых
категорий"""

# Объединение активности пользователя и любимых категорий

user_profile = user_activity.merge(
    top_categories,
    on="user_id",
    how="left"
)

print("Профиль пользователя успешно дополнен любимыми
категориями.")
display(user_profile.head(10))

"""3.3.5 Расчёт предпочтений пользователей по конкретным
блюдам"""

# Расчёт предпочтений пользователей по конкретным блюдам

user_item_preferences = (
    orders_df
    .groupby(["user_id", "item_id", "item_name"])
    .agg(
        item_quantity=("quantity", "sum"),
        item_orders=("order_id", "nunique")
    )
    .reset_index()
)

user_item_preferences = user_item_preferences.sort_values(
    by=["user_id", "item_quantity"],
    ascending=[True, False]
)

print("Предпочтения пользователей по блюдам успешно
рассчитаны.")
display(user_item_preferences.head(15))

"""3.3.6 Топ-5 любимых блюд для каждого пользователя"""

# Топ-5 любимых блюд для каждого пользователя

top_items = (
    user_item_preferences
    .groupby("user_id")
    .head(5)
    .groupby("user_id")["item_name"]
    .apply(lambda x: ", ".join(x))

```

```

        .reset_index()
    )

top_items = top_items.rename(
    columns={"item_name": "favorite_items"}
)

print("Топ-5 блюд для каждого пользователя сформирован.")
display(top_items.head(5))

"""3.3.7 Добавление любимых блюд в профиль пользователя"""

# Добавление любимых блюд в профиль пользователя

user_profile = user_profile.merge(
    top_items,
    on="user_id",
    how="left"
)

print("Профиль пользователя дополнен любимыми блюдами.")
display(user_profile.head(10))

"""3.3.8 Классификация пользователей по объёму истории заказов"""

# Классификация пользователей по объёму истории заказов

def define_user_group(orders_count):

    if orders_count <= 2:
        return "Новый пользователь"

    elif orders_count <= 5:
        return "Пользователь с малой историей"

    else:
        return "Активный пользователь"

user_profile["user_group"] = user_profile[
    "orders_count"
].apply(define_user_group)

print("Классификация пользователей успешно выполнена.")

display(
    user_profile[
        [
            "user_id",
            "orders_count",
            "user_group"
        ]
    ].head(10)

```

```

)

"""3.3.9 Распределение пользователей по группам"""

# Распределение пользователей по группам

user_group_distribution = (
    user_profile["user_group"]
    .value_counts()
    .reset_index()
)

user_group_distribution.columns = [
    "user_group",
    "users_count"
]

print("Распределение пользователей по группам:")

display(user_group_distribution)

"""3.3.10 Визуализация распределения пользователей"""

# Визуализация распределения пользователей

import matplotlib.pyplot as plt

group_counts = user_profile[
    "user_group"
].value_counts()

plt.figure(figsize=(9, 5))

group_counts.plot(kind="bar")

plt.title(
    "Распределение пользователей по уровню активности"
)

plt.xlabel("Группа пользователей")
plt.ylabel("Количество пользователей")

plt.xticks(rotation=10)

plt.grid(axis="y")

plt.show()

from itertools import combinations
from collections import Counter

# Формирование корзин заказов

order_baskets = (

```

```

        orders_df
        .groupby("order_id")["item_name"]
        .apply(list)
        .reset_index()
    )

print("Корзины заказов успешно сформированы.")
display(order_baskets.head(10))

# Формирование пар блюд, которые встречались в одном заказе

pair_counter = Counter()

for basket in order_baskets["item_name"]:
    unique_items = sorted(set(basket))

    for item_pair in combinations(unique_items, 2):
        pair_counter[item_pair] += 1

print("Пары совместно приобретаемых блюд успешно рассчитаны.")

# Преобразование результата в DataFrame

co_purchase_df = pd.DataFrame(
    [
        {
            "item_1": pair[0],
            "item_2": pair[1],
            "purchase_count": count
        }
        for pair, count in pair_counter.items()
    ]
)

co_purchase_df = co_purchase_df.sort_values(
    by="purchase_count",
    ascending=False
).reset_index(drop=True)

print("Топ-10 наиболее частых совместных покупок:")
display(co_purchase_df.head(10))

# Сохранение датасета совместных покупок

co_purchase_df.to_csv(
    "co_purchase_dataset.csv",
    index=False,
    encoding="utf-8-sig"
)

print("Датасет совместных покупок сохранён.")

"""3.3.11 Пример полного случайного пользовательского профиля"""

```

```

# Пример полного пользовательского профиля

import random

# Случайный пользователь из датасета
example_user_id = random.choice(
    user_profile["user_id"].unique()
)

example_profile = user_profile[
    user_profile["user_id"] == example_user_id
]

print(f"Профиль пользователя {example_user_id}:")

display(
    example_profile[
        [
            "user_id",
            "orders_count",
            "items_count",
            "unique_items",
            "favorite_categories",
            "favorite_items",
            "user_group"
        ]
    ]
)

"""**3.4 ОБУЧЕНИЕ МОДЕЛЕЙ**

**3.4.1 Content-Based модель рекомендаций**
"""

# Формирование индексов блюд

indices = pd.Series(
    menu_df.index,
    index=menu_df["name"]
).drop_duplicates()

print("Количество блюд в индексе:")
print(len(indices))

from sklearn.metrics.pairwise import cosine_similarity

# Расчёт матрицы сходства между блюдами

cosine_sim = cosine_similarity(tfidf_matrix)

print("Матрица сходства content-based модели рассчитана.")
print("Размер матрицы:")
print(cosine_sim.shape)

```

```

# Функция поиска похожих блюд

def get_similar_items(item_name, top_n=5):

    if item_name not in indices:
        print("Блюдо не найдено в датасете.")
        return pd.DataFrame()

    item_idx = indices[item_name]

    similarity_scores = list(enumerate(cosine_sim[item_idx]))

    similarity_scores = sorted(
        similarity_scores,
        key=lambda x: x[1],
        reverse=True
    )

    similarity_scores = similarity_scores[1:top_n + 1]

    item_indices = [score[0] for score in similarity_scores]
    similarity_values = [score[1] for score in
similarity_scores]

    recommendations = menu_df.iloc[item_indices][
        ["item_id", "name", "category", "cuisine", "price"]
    ].copy()

    recommendations["similarity"] = similarity_values

    return recommendations

# Получение рекомендаций для выбранного блюда

selected_item = "Капучино"

content_based_recommendations = get_similar_items(
    selected_item,
    top_n=5
)

print(f"Рекомендации для блюда: {selected_item}")
display(content_based_recommendations)

# Случайный выбор блюд для тестирования content-based модели

test_items = menu_df["name"].sample(
    n=5,
    #random_state=42
).tolist()

# Проверка работы модели на случайных примерах

for item in test_items:

```

```

print(f"\nРекомендации для блюда: {item}")

recommendations = get_similar_items(
    item,
    top_n=5
)

display(recommendations)

# Формирование рекомендаций для нескольких блюд

content_based_results = []

for item in test_items:

    recommendations = get_similar_items(
        item,
        top_n=5
    )

    for _, row in recommendations.iterrows():
        content_based_results.append(
            {
                "source_item": item,
                "recommended_item": row["name"],
                "category": row["category"],
                "cuisine": row["cuisine"],
                "price": row["price"],
                "similarity": row["similarity"]
            }
        )

content_based_results_df = pd.DataFrame(content_based_results)

print("Результаты тестирования content-based модели:")
display(content_based_results_df)

# Сохранение результатов тестирования

content_based_results_df.to_csv(
    "content_based_test_results.csv",
    index=False
)

print("Результаты тестирования content-based модели сохранены.")

"""**3.4.2 Модель поиска сопутствующих товаров методом kNN**"""

# Построение матрицы "заказ - блюдо"

order_item_matrix = orders_df.pivot_table(
    index="order_id",
    columns="item_name",

```

```

        values="quantity",
        aggfunc="sum",
        fill_value=0
    )

print("Размерность матрицы заказов и блюд:")
print(order_item_matrix.shape)

display(order_item_matrix.head())

# Подготовка item-item матрицы

item_order_matrix = order_item_matrix.T

print("Размерность item-item матрицы:")
print(item_order_matrix.shape)

from sklearn.neighbors import NearestNeighbors

# Обучение модели kNN

knn_model = NearestNeighbors(
    metric="cosine",
    algorithm="brute"
)

knn_model.fit(item_order_matrix)

print("Модель kNN успешно обучена.")

# Функция поиска сопутствующих товаров

def get_knn_recommendations(item_name, top_n=5):

    if item_name not in item_order_matrix.index:
        print("Блюдо не найдено в матрице заказов.")
        return pd.DataFrame()

    item_vector = item_order_matrix.loc[[item_name]]

    distances, indices = knn_model.kneighbors(
        item_vector,
        n_neighbors=top_n + 1
    )

    recommended_items = []

    for idx, distance in zip(indices[0][1:], distances[0][1:]):

        recommended_items.append(
            {
                "source_item": item_name,
                "recommended_item":
item_order_matrix.index[idx],

```

```

        "similarity": round(1 - distance, 3)
    }
)

recommendations_df = pd.DataFrame(recommended_items)

return recommendations_df

# Получение рекомендаций для выбранного блюда
selected_item = "Рыба с рисом"

knn_recommendations = get_knn_recommendations(
    selected_item,
    top_n=5
)

print(f"Сопутствующие рекомендации для блюда: {selected_item}")
display(knn_recommendations)

# Случайный выбор блюд для тестирования kNN-модели
test_items_knn = item_order_matrix.index.to_series().sample(
    n=5,
    #random_state=42
).tolist()

# Тестирование модели на нескольких блюдах
knn_results = []

for item in test_items_knn:
    recommendations = get_knn_recommendations(
        item,
        top_n=5
    )

    for _, row in recommendations.iterrows():
        knn_results.append(row.to_dict())

knn_results_df = pd.DataFrame(knn_results)

print("Результаты тестирования kNN-модели:")
display(knn_results_df)

# Сохранение результатов kNN-рекомендаций
knn_results_df.to_csv(
    "knn_test_results.csv",
    index=False
)

print("Результаты тестирования kNN-модели сохранены.")

```

```
"""**3.4.3 Модель коллаборативной фильтрации на основе SVD**"""
```

```
# Установка библиотеки Surprise
```

```
#!pip install "numpy<2" scikit-surprise --force-reinstall
```

```
from surprise import Dataset, Reader, SVD
from surprise.model_selection import train_test_split
from surprise import accuracy
```

```
# Подготовка данных для SVD
```

```
reader = Reader(
    rating_scale=(1, 5)
)
```

```
data = Dataset.load_from_df(
    ratings_df[["user_id", "item_id", "rating"]],
    reader
)
```

```
print("Данные для SVD подготовлены.")
```

```
# Разделение данных на обучающую и тестовую выборки
```

```
trainset, testset = train_test_split(
    data,
    test_size=0.2,
    #random_state=42
)
```

```
print("Train/test split выполнен.")
```

```
# Обучение SVD-модели
```

```
svd_model = SVD(
    n_factors=50,
    n_epochs=30,
    #random_state=42
)
```

```
svd_model.fit(trainset)
```

```
print("SVD-модель успешно обучена.")
```

```
# Получение предсказаний на тестовой выборке
```

```
svd_predictions = svd_model.test(testset)
```

```
print("Количество предсказаний:")
print(len(svd_predictions))
```

```

# Просмотр примеров предсказаний на тестовой выборке

predictions_df = pd.DataFrame(
    [
        {
            "user_id": pred.uid,
            "item_id": pred.iid,
            "real_rating": pred.r_ui,
            "predicted_rating": round(pred.est, 3)
        }
        for pred in svd_predictions[:100]
    ]
)

display(predictions_df)

# Функция формирования персональных рекомендаций

def get_svd_recommendations(user_id, top_n=5):

    user_items = ratings_df[
        ratings_df["user_id"] == user_id
    ][["item_id"].unique()

    all_items = menu_df["item_id"].unique()

    items_to_predict = [
        item for item in all_items
        if item not in user_items
    ]

    predictions = []

    for item_id in items_to_predict:

        predicted_rating = svd_model.predict(
            user_id,
            item_id
        ).est

        predictions.append(
            {
                "user_id": user_id,
                "item_id": item_id,
                "predicted_rating": round(predicted_rating, 3)
            }
        )

    recommendations_df = pd.DataFrame(predictions)

    recommendations_df = recommendations_df.sort_values(
        by="predicted_rating",
        ascending=False
    ).head(top_n)

```

```

    recommendations_df = recommendations_df.merge(
        menu_df[["item_id", "name", "category", "price"]],
        on="item_id",
        how="left"
    )

    return recommendations_df

# Получение персональных рекомендаций для случайного
пользователя

selected_user = ratings_df["user_id"].sample(
    n=1,
    #random_state=42
).iloc[0]

svd_recommendations = get_svd_recommendations(
    selected_user,
    top_n=5
)

print(f"Персональные рекомендации для пользователя:
{selected_user}")
display(svd_recommendations)

# Тестирование SVD-модели на нескольких пользователях

test_users = ratings_df["user_id"].drop_duplicates().sample(
    n=5,
).tolist()

svd_results = []

for user in test_users:

    recommendations = get_svd_recommendations(
        user,
        top_n=5
    )

    for _, row in recommendations.iterrows():
        svd_results.append(row.to_dict())

svd_results_df = pd.DataFrame(svd_results)

print("Результаты тестирования SVD-модели:")
display(svd_results_df)

# Сохранение результатов SVD-рекомендаций

svd_results_df.to_csv(
    "svd_test_results.csv",
    index=False

```

```

)

print("Результаты тестирования SVD-модели сохранены.")

"""**Гибридная рекомендательная система**"""

# Функция гибридной рекомендации

def get_hybrid_recommendations(
    user_id=None,
    item_name=None,
    scenario="personal",
    top_n=5
):

    if scenario == "cold_start" and item_name is not None:

        recommendations = get_similar_items(
            item_name,
            top_n=top_n
        )

        recommendations["model"] = "Content-Based"

        return recommendations

    elif scenario == "cross_sale" and item_name is not None:

        recommendations = get_knn_recommendations(
            item_name,
            top_n=top_n
        )

        recommendations["model"] = "kNN"

        return recommendations

    elif scenario == "personal" and user_id is not None:

        recommendations = get_svd_recommendations(
            user_id,
            top_n=top_n
        )

        recommendations["model"] = "SVD"

        return recommendations

    else:

        print("Недостаточно данных для формирования рекомендаций.")
        return pd.DataFrame()

```

```

# Проверка сценария cold start

hybrid_cold_start = get_hybrid_recommendations(
    item_name="Капучино",
    scenario="cold_start",
    top_n=5
)

print("Гибридная рекомендация: cold start")
display(hybrid_cold_start)

# Проверка сценария kNN

hybrid_cross_sale = get_hybrid_recommendations(
    item_name="Капучино",
    scenario="cross_sale",
    top_n=5
)

print("Гибридная рекомендация: cross-sale")
display(hybrid_cross_sale)

# Проверка сценария персональных рекомендаций

selected_user = ratings_df["user_id"].sample(
    n=1,
    random_state=42
).iloc[0]

hybrid_personal = get_hybrid_recommendations(
    user_id=selected_user,
    scenario="personal",
    top_n=5
)

print(f"Гибридная рекомендация: персональные рекомендации для пользователя {selected_user}")
display(hybrid_personal)

# Объединение результатов тестирования гибридной системы

hybrid_results = pd.concat(
    [
        hybrid_cold_start,
        hybrid_cross_sale,
        hybrid_personal
    ],
    ignore_index=True
)

print("Результаты работы гибридной рекомендательной системы:")
display(hybrid_results)

# Сохранение результатов гибридной системы

```

```

hybrid_results.to_csv(
    "hybrid_recommendations.csv",
    index=False
)

print("Результаты гибридной рекомендательной системы
сохранены.")

# Количественные характеристики данных

users_count = ratings_df["user_id"].nunique()

items_count = menu_df["item_id"].nunique()

orders_count = orders_df["order_id"].nunique()

min_rating = ratings_df["rating"].min()
max_rating = ratings_df["rating"].max()

print("Количество пользователей:", users_count)
print("Количество блюд:", items_count)
print("Количество заказов:", orders_count)
print("Диапазон рейтингов:", f"{min_rating}-{max_rating}")

"""**4.4 РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНОГО ИССЛЕДОВАНИЯ МОДЕЛЕЙ**

4.4.1 Исследование проблемы холодного старта
"""

# Новый пользователь без истории заказов

new_user_id = 125

print(f"Новый пользователь: {new_user_id}")

if new_user_id in ratings_df["user_id"].unique():
    print("Пользователь имеет историю заказов")
else:
    print("Пользователь НЕ имеет историю заказов")

# Попытка получить рекомендации SVD для нового пользователя

svd_cold_start = get_svd_recommendations(
    new_user_id,
    top_n=5
)

print(f"SVD-рекомендации для нового пользователя
{new_user_id}:")
display(svd_cold_start)

# Content-based рекомендации для нового пользователя на основе
выбранного блюда

```

```

selected_item = "Капучино"

cb_cold_start = get_similar_items(
    selected_item,
    top_n=5
)

print(f"Content-based рекомендации для нового пользователя
{new_user_id} на основе блюда: {selected_item}")
display(cb_cold_start)

# Поиск пользователей с малой историей заказов

user_interactions = ratings_df.groupby("user_id").agg(
    interactions_count=("item_id", "count")
).reset_index()

low_history_users = user_interactions[
    user_interactions["interactions_count"] <= 2
]

print("Пользователи с малой историей:")
display(low_history_users.head())

# Случайный выбор пользователя с малой историей

low_history_user = low_history_users[
    "user_id"
].sample(1).iloc[0]

print("Случайно выбранный пользователь с малой историей:")
print(low_history_user)

# SVD-рекомендации для пользователя с малой историей

svd_low_history = get_svd_recommendations(
    low_history_user,
    top_n=5
)

print(f"SVD-рекомендации для пользователя с малой историей:
{low_history_user}")
display(svd_low_history)

# Просмотр истории пользователя с малой историей

low_history_items = ratings_df[
    ratings_df["user_id"] == low_history_user
].merge(
    menu_df[["item_id", "name", "category", "price"]],
    on="item_id",
    how="left"
)

```

```

print("История пользователя с малой историей:")
display(low_history_items)

# Content-based рекомендации на основе последнего блюда
пользователя

last_item_name = low_history_items["name"].iloc[0]

cb_low_history = get_similar_items(
    last_item_name,
    top_n=5
)

print(f"Content-based рекомендации на основе блюда:
{last_item_name}")
display(cb_low_history)

# Сводная таблица результатов холодного старта

cold_start_summary = pd.DataFrame(
    [
        {
            "scenario": "Новый пользователь без истории",
            "model": "SVD",
            "basis": "Нет истории взаимодействий",
            "result": "Персонализация ограничена"
        },
        {
            "scenario": "Новый пользователь без истории",
            "model": "Content-based",
            "basis": selected_item,
            "result": "Сформированы рекомендации похожих блюд"
        },
        {
            "scenario": "Пользователь с малой историей",
            "model": "SVD",
            "basis": f"user_id = {low_history_user}",
            "result": "Рекомендации сформированы при
ограниченной истории"
        },
        {
            "scenario": "Пользователь с малой историей",
            "model": "Content-based",
            "basis": last_item_name,
            "result": "Сформированы рекомендации на основе
выбранного блюда"
        }
    ]
)

display(cold_start_summary)

# Сохранение результатов исследования холодного старта

```

```

cold_start_summary.to_csv(
    "cold_start_summary.csv",
    index=False
)

print("Результаты исследования холодного старта сохранены.")

"""4.4.2 Исследование зависимости качества рекомендаций SVD от
объема входных данных"""

# Функция обучения и оценки SVD-модели

def evaluate_svd_on_data(dataframe, data_part):

    reader = Reader(
        rating_scale=(1, 5)
    )

    data = Dataset.load_from_df(
        dataframe[["user_id", "item_id", "rating"]],
        reader
    )

    trainset, testset = train_test_split(
        data,
        test_size=0.2,
        random_state=42
    )

    model = SVD(
        n_factors=50,
        n_epochs=30,
        random_state=42
    )

    model.fit(trainset)

    predictions = model.test(testset)

    rmse_value = accuracy.rmse(
        predictions,
        verbose=False
    )

    mae_value = accuracy.mae(
        predictions,
        verbose=False
    )

    return {
        "data_part": data_part,
        "records_count": len(dataframe),
        "rmse": round(rmse_value, 3),

```

```

        "mae": round(mae_value, 3)
    }

# Формирование выборок разного объёма

users_30 = ratings_df["user_id"].drop_duplicates().sample(
    frac=0.3,
    #random_state=42
)

ratings_30 = ratings_df[
    ratings_df["user_id"].isin(users_30)
]

users_60 = ratings_df["user_id"].drop_duplicates().sample(
    frac=0.6,
    #random_state=42
)

ratings_60 = ratings_df[
    ratings_df["user_id"].isin(users_60)
]

ratings_100 = ratings_df.copy()

print("Размеры выборок:")
print("30%:", len(ratings_30))
print("60%:", len(ratings_60))
print("100%:", len(ratings_100))

# Обучение и оценка SVD-модели на выборках разного объёма

svd_data_volume_results = []

for dataframe, data_part in [
    (ratings_30, "30%"),
    (ratings_60, "60%"),
    (ratings_100, "100%")
]:

    result = evaluate_svd_on_data(
        dataframe,
        data_part
    )

    svd_data_volume_results.append(result)

svd_data_volume_df = pd.DataFrame(svd_data_volume_results)

print("Результаты исследования влияния объёма данных:")
display(svd_data_volume_df)

# Сохранение результатов исследования

```

```

svd_data_volume_df.to_csv(
    "svd_data_volume_results.csv",
    index=False
)

print("Результаты исследования сохранены.")

import matplotlib.pyplot as plt

# Построение графика изменения RMSE

plt.figure(figsize=(8, 5))

plt.plot(
    svd_data_volume_df["data_part"],
    svd_data_volume_df["rmse"],
    marker="o",
    label="RMSE"
)

plt.plot(
    svd_data_volume_df["data_part"],
    svd_data_volume_df["mae"],
    marker="o",
    label="MAE"
)

plt.title("Влияние объёма данных на качество SVD-модели")
plt.xlabel("Объём данных")
plt.ylabel("Значение ошибки")
plt.legend()
plt.grid(True)

plt.show()

```

ПРИЛОЖЕНИЕ Б

(Обязательное)

Рекомендательная система автоматизации работы заведений
общественного питания

Презентация

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА АВТОМАТИЗАЦИИ РАБОТЫ ЗАВЕДЕНИЙ ОБЩЕСТВЕННОГО ПИТАНИЯ

Студент гр. 24ВВИм1: **Мартынов Н.В.**

Руководитель: к.т.н, доцент кафедры ВТ **Карамышева Н.С.**

Пенза 2026

Актуальность проекта

- **Тренд:** современный рынок требует быстрого и персонального обслуживания клиентов
- **Контраст:** крупные сети используют дорогие ИИ-платформы, которые недоступны малому бизнесу из-за высокой стоимости и зависимости от больших данных
- **Проблема:** ручной подбор блюд сотрудниками заведения работает только при малом потоке гостей
- **Порог автоматизации:** при потоке от 100 чеков в день ручной подход создает очереди
- **Решение:** разработка собственной автономной рекомендательной системы для заведений общественного питания малого масштаба



Цель, задачи и научная гипотеза

Цель проекта: разработка и исследование рекомендательной системы автоматизации для заведений общественного питания малого сегмента.

Научная гипотеза:

Использование гибридной архитектуры позволит повысить качество рекомендаций для различных сценариев обслуживания, обеспечивая высокую точность без привлечения больших объемов данных и затрат на внедрение сложных систем.

Задачи исследования:

- 1) Анализ существующих решений на рынке.
- 2) Сбор, предобработка и векторизация данных.
- 3) Обучение моделей и оценка качества рекомендаций.
- 4) Проведение вычислительных экспериментов.
- 5) Интеграция полученных результатов в программный комплекс.

Анализ существующих решений

Критерии сравнения	Облачные ИИ-платформы (Яндекс Еда, Amazon)	Локальные POS-модули (r_keeper Kiosk Pro)	Разрабатываемая система
Стоимость внедрения	Высокая (недоступно малым кафе)	Доступная (встроено в терминал)	Низкая (для малого бизнеса)
Зависимость от данных	Требуются терабайты истории (Big Data)	Не требуются (настройка вручную)	Минимальная (решено синтетически)
Автономность работы	Нет (только постоянный онлайн)	Да (работает локально)	Да (локальный POS-терминал)
Автоматическое обучение	Да (ИИ переобучается сам)	Нет (жесткие ручные сценарии)	Да (динамический подбор)

Архитектура гибридной системы



Модуль рекомендаций (SVD)

Алгоритм SVD решает задачу **коллаборативной фильтрации**, находя скрытые закономерности в поведении групп клиентов со схожими вкусами.

Матричное разложение:

$$R \approx P \times Q^T$$

Расчет прогнозной оценки пользователя для блюда:

$$\hat{r}_{u,i} = \mu + b_u + b_i + p_u \cdot q_i^T$$

Формула неявного рейтинга: $\text{Rating} = \min(5, 1 + k \cdot N_{u,i})$

ПРИМЕР РАБОТЫ МОДУЛЯ СТАРЫЙ КЛИЕНТ

Авторизованный гость: Клиент #14 (Постоянный)

История заказов: Зеленый чай, Лагман, Рис с овощами, Борщ с курицей, Рис отварной.

Рекомендация: «Плов с курицей»
Прогноз интереса: 4.8 / 5.0

Контентный модуль (Content-Based)

Обеспечивает подбор похожих позиций меню для новых гостей без истории заказов на базе состава блюд, решая **проблему «холодного старта»**.

Мера косинусного сходства векторов (TF-IDF):

$$\text{Cosine Similarity} (A, B) = \frac{A \cdot B}{|A| |B|}$$

Где A и B - векторизованные ингредиентные профили блюд из базы данных.

ПРИМЕР РАБОТЫ МОДУЛЯ

НОВЫЙ ПОЛЬЗОВАТЕЛЬ

Активная позиция в чеке:

Капучино (300 мл)

Состав блюда:

Молоко коровье, Эспрессо кофейный

Рекомендация №1:

«Латте» (коэффициент сходства: 0.85)

Рекомендация №2:

«Кофе Раф» (коэффициент сходства: 0.78)

Ассоциативный модуль (kNN)

Ориентирован на сопутствующие кросс-продажи (Cross-Sell) на этапе оформления заказа.

 **Поиск соседей:** Алгоритм ищет k товаров, наиболее часто встречающихся совместно в истории чеков.

 **Метрика близости:** косинусное расстояние совместных покупок между объектами.

Евклидова метрика вычисления расстояния между «соседями»:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Где: x и y – векторы признаков объектов
n – количество признаков
d(x, y) – расстояние между объектами

ПРИМЕР РАБОТЫ МОДУЛЯ

ФОРМИРОВАНИЕ КОРЗИНЫ

Позиция в корзине:

Мясной сет №1

Сумма заказа: 890 руб.

Рекомендовано к блюду:

«Соус Барбекю» (совместных покупок: 83%)

Рекомендовано к блюду:

«Картофель по-деревенски» (совместных покупок: 71%)

Сбор и подготовка данных

Источник 1

ЛОКАЛЬНАЯ БД ЗАВЕДЕНИЯ
1 000 чеков

Исторические транзакции из локальной базы данных заведения

Источник 2

ИМИТАЦИОННАЯ МОДЕЛЬ
4 211 чеков

Генерация данных для решения проблемы высокой разреженности матрицы

ИТОГОВЫЙ ДАТАСЕТ
5 211 чеков

Достаточный массив данных для стабильного обучения моделей без переобучения

Набор данных	Структура данных датасета	Используемая модель
menu_dataset	Триплеты (User_ID, Item_ID, Rating) на базе частоты покупок блюда	SVD
orders_dataset	Векторизованные признаки (TF-IDF) ингредиентного состава блюд	Content-based
ratings_dataset	Матрица смежности совместных появлений позиций меню в чеках	kNN

Проблема с «холодного старта» для SVD

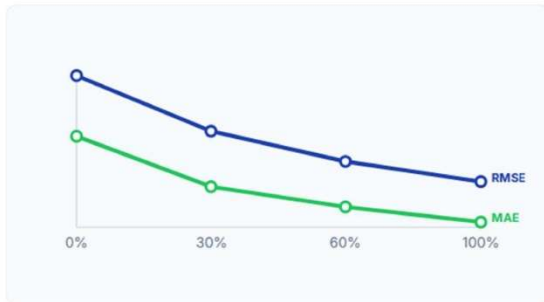
Зависимость качества рекомендаций SVD от объема обучающей выборки:

Объем данных	Тип выборки	Погрешность RMSE	Погрешность MAE
0%	Проблема «холодного старта»	1.520	1.210
30%	Только реальные чеки	1.082	0.865
60%	Частичное синтетическое расширение	0.945	0.750
100%	Гибридный набор (Реальные + Синтетические)	0.841	0.635

Вывод: имитационное моделирование потребительского поведения позволило **снизить ошибку прогноза RMSE на 22.2%** относительно выборки реальных чеков.

Визуализация точности SVD и оценка других моделей

Динамика снижения ошибок:



Оценка точности работы других моделей:

В отличие от SVD, эффективность ассоциативного и контентного модулей оценивалась иначе:

Метод оценки Content-based и kNN:

Использован **качественный подход**

Релевантность выдачи («Плов к мясному сету» или «Латте к капучино») подтверждена ручным тестированием на соответствие логике меню заведения.

Технический итог:

Среднее время формирования рекомендации (Latency) составило **17 мс**.

Заключение

Научные результаты:

- Разработана трехмодульная гибридная рекомендательная система.
- Внедрена методика расчета неявного рейтинга.
- Решена проблема «холодного старта».

Практическая значимость:

- Система готова для внедрения в работу существующей программы.
- Имеет простой и понятный интерфейс работы.
- Повышает лояльность клиентов заведений за счет индивидуального подхода.
- Способна повысить доход заведений.

Список научных публикаций

Результаты исследования представлены в следующих работах:



**[Статья 1]: Система автоматизации работы заведений общественного питания
малого масштаба**

РИНЦ

«Новые информационные технологии и системы» (НИТиС-2024)», г. Пенза, ПГУ.



**[Статья 2]: Система автоматизации работы заведений общественного
питания малого масштаба**

РИНЦ

Конкурс студенческих научных работ «IT-Орен», г. Рязань, 2025.



**[Статья 3]: Использование гибридных архитектур в рекомендательных
системах**

РИНЦ

Вестник ПГУ-2026, г. Пенза.

СПАСИБО ЗА ВНИМАНИЕ!

Готов ответить на ваши вопросы по теме научно-практической работы!