

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: «Вычислительные машины, комплексы, системы и сети»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПО
ФОРМИРОВАНИЮ МЕНЮ ДЛЯ ПИТАНИЯ
В ШКОЛЬНЫХ СТОЛОВЫХ**

Студент

Саф 26.05.26 Сафронов Денис Витальевич
(подпись, дата) (ФИО полностью)

Руководитель

Токарев 26.05.26 Токарев А.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

Бычков 26.05.26 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент

главный специалист 26.05.26 Васильков А.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой

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

Работа защищена с отметкой отлично (протокол заседания ГЭК от

04 июня 2026 № 9)

Секретарь ГЭК

Гурин Е.И.
(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 гр. 24ВВиМ1 факультета ВТ Сафронов Денис Витальевич
2. Тема работы «Рекомендательная система по формированию меню для питания в школьных столовых»
Тема утверждена приказом ПГУ № 920/0 от "20" сентября 2024г.
3. Руководитель магистерской диссертации к.т.н., доцент Токарев Андрей Николаевич
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: автоматизация процессов формирования меню школьной столовой, учет потребительского спроса в школьных столовых.
Разработать рекомендательную систему на основе правил и провести её сравнение с методами машинного обучения.
Исходные данные: список заказов, список обучающихся с информацией о возрасте и наличии ограничений, список блюд, данные о нутриентном составе блюд
5. Перечень подлежащих разработке вопросов
 1. Анализ предметной области
 2. Разработка рекомендательной системы формирования школьного меню
 3. Тестирование системы и сравнительный анализ с альтернативными подходами
 4. Анализ и интерпретация полученных результатов

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	15%	21.09.2024- 21.11.2024	
2	Разработка алгоритма	15%	22.11.2024- 22.01.2025	
3	Проектирование структуры	15%	23.01.2025- 23.04.2025	
4	Разработка компонентов	40%	24.04.2025- 29.01.2026	
5	Отладка рекомендательной системы	10%	30.01.2026- 30.03.2026	
6	Оформление выпускной работы	10%	31.03.2026- 31.05.2026	

Дата выдачи задания " 20 " 09 2024 г.

Руководитель магистерской диссертации



(подпись, дата)

Задание к исполнению принял студент



(подпись, дата)

Работу к защите допустить

Декан факультета



А.Р. Ресов

(подпись, дата)

02.06.2020

Реферат

Пояснительная записка содержит 86 листов, 9 рисунков, 6 таблиц, 8 источников, 6 формул, 2 приложений

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, АНАЛИЗ ЗАПРОСОВ, МАШИННОЕ ОБУЧЕНИЕ. ПИТАНИЕ, САНПИН, АЛГОРИТМ, ФОРМИРОВАНИЕ МЕНЮ, AI

Цель работы – проанализировать существующие рекомендательные системы и оценить возможность их внедрения в процессы организации питания в школьных столовых.

Объектом разработки является рекомендательная система, выполняющая расчет необходимого количества порций с учетом типа блюда, его калорийности и бжу. Расчет выполняется на основании анализа заказов, возраста и индивидуальных ограничений каждого ученика.

Предметом исследования является рекомендательная система, позволяющая автоматизировать процесс формирования персонализированного школьного меню.

Задачи исследования:

- провести обзор существующих решений автоматизации столовых,
- выявить критерии применимости рекомендательных систем для школьных столовых,
- проанализировать требования СанПиН к организации питания в образовательных учреждениях,
- разработать рекомендательную систему с учетом требований СанПиН,
- протестировать результат работы системы.

В результате проведенной работы разработана рекомендательная система, полностью соответствующая поставленной цели данной работы.

					ПГУ1.090401.19.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Пояснительная записка «Рекомендательная система по формированию меню для питания в школьных столовых»	Лит.	Лист	Листов
Разраб.		Сафронов Д.В.	<i>Саф</i>	26.05.26				
Провер.		Токарев А.Н.	<i>Ток</i>	26.05.26			2	86
Н. Контр.		Бычков А.С.	<i>Б</i>	26.05.26		ФВТ, каф. ВТ гр. 24ВВИМ1		
Утверд.		Митрохин М.А.	<i>М</i>	26.05.26				

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	4
ВВЕДЕНИЕ	5
1 Анализ предметной области	8
1.1 Обзор цифровых решений для школьного питания	8
1.2 Нормы СанПиН	10
1.3 Постановка задачи	14
2 Требования к системе	17
2.1 Требования к данным	17
2.2 Требования к алгоритму	19
3 Разработка рекомендательной системы	21
3.1 Описание алгоритма	21
3.2 Выбор метода фильтрации	22
3.3 Механизм формирования меню	24
4 Реализация системы	25
4.1 Подготовка данных	25
4.2 Архитектура системы	31
4.2 Реализация алгоритма	35
5 Эксперимент и оценка	41
5.1 Сравнение со стандартным подходом	43
5.2 Методика и результаты сравнения текущего алгоритма и ML –	
подходов	44
5.2.1 Разработанный алгоритм	45
5.2.2 CatBoostRanker	46
5.2.3 XGBRanker	46
5.2.4 Результаты эксперимента	47
ЗАКЛЮЧЕНИЕ	53
СПИСОК ИСТОЧНИКОВ	55
ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ	56
ПРИЛОЖЕНИЕ Б – ПРЕЗЕНТАЦИЯ	80

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В данной работе применены следующие обозначения

БЖУ – белки, жиры, углеводы

ПО – программное обеспечение

СанПиН – санитарные правила и нормы

API (Application Programming Interface) – программный интерфейс

позволяющий приложениям взаимодействовать между собой

JSON – текстовый формат для хранения и передачи структурированных данных

ВВЕДЕНИЕ

Организации питания в школьных столовых играет важную роль в обеспечении здоровья и развития детей. По этому задача формирования меню является одной из самых важных на сегодняшний день. Правильное меню должно учитывать возрастные и физиологические особенности ученика, нормы калорийности, сбалансированность рациона и требования СанПиН. Традиционные методы составления меню часто основываются на ручной работе, что приводит к низкой эффективности и не учитывает индивидуальных предпочтений школьников. Это снижает гибкость и адаптивность предложений, а также увеличивает нагрузку на персонал столовой.

С развитием информационных технологий появляется возможность автоматизировать процесс формирования меню с помощью рекомендательных систем[1]. Их работа основывается на алгоритмах машинного обучения и анализа данных, что позволяет учитывать множество факторов, такие как предпочтения учеников, индивидуальные ограничения, возраст, состав блюд, калорийность, БЖУ, финансовые ограничения и санитарные нормы. Использование рекомендательных систем помогает решать задачи персонала школьных столовых, это способствует улучшению качества питания, снижению пищевых отходов и повышению удовлетворённости учеников. Кроме того, такая система обеспечивает удобство использования[2]. Ученики и родители могут влиять на составление меню с помощью системы рейтинга и просматривать историю заказов. Работники столовой также могут отслеживать и контролировать количество необходимых продуктов для оптимального использования.

В процессе выполнения выпускной квалификационной работы необходимо автоматизировать процесс формирования меню школьных столовых. Рекомендательная система направлена на автоматизацию формирования рационов с учетом ограничений, статистики заказов и индивидуальных предпочтений. Система позволит сократить временные и

трудовые затраты на составление меню и повысить качество обслуживания учащихся.

Повышение эффективности достигается за счет применения автоматизированных алгоритмов подборов блюд, основанных на анализе пользовательских данных и гибких правилах фильтрации. Расчет необходимых порций выполняется по типу блюда, калориям, БЖУ в соответствии с нормами СанПин[3]. Также проводится анализ заказов студентов. На основании этого формируется персонализированное меню для школьников, что делает систему универсальной и востребованной.

Объектом исследования является процесс организации питания в школьных столовых.

Предметом исследования является рекомендательная система, позволяющая автоматизировать процесс формирования персонализированного школьного меню.

Требования СанПиН[3] к организации школьного питания строго регламентированы, повсеместно контролируются и в целом положительно влияют на здоровье учащихся. Однако, как показало исследование Столь А.В., посвящённое удовлетворённости школьников питанием, основными проблемами остаются температура и вкус подаваемых блюд. Если с сентября 2025 года введено обязательное требование по установке термометров для контроля температуры, то проблема вкусовых качеств еды до сих пор не решена. Между тем, именно вкус напрямую влияет на процент фактического потребления пищи в школе, что делает данную задачу актуальной для дальнейшей проработки.

Цель исследования: Проанализировать существующие рекомендательные системы и оценить возможность их внедрения в процессы организации питания в школьных столовых.

Задачи исследования:

- провести обзор существующих решений автоматизации столовых,

- выявить критерии применимости рекомендательных систем для школьных столовых,
- проанализировать требования СанПиН к организации питания в образовательных учреждениях,
- разработать рекомендательную систему с учетом требований СанПиН,
- протестировать результат работы системы.

Научная новизна заключается в систематизации критериев применимости рекомендательных систем в школьных столовых и выявлении основных ограничений (СанПиН, возрастные особенности, пищевые предпочтения), которые должны учитываться при их внедрении.

Практическая значимость исследования заключается в разработке программного прототипа рекомендательной системы на Python, который может быть использован сотрудниками школьных столовых для автоматизации формирования меню, расчёта порций по калориям и БЖУ, а также учёта индивидуальных предпочтений учащихся.

1 Анализ предметной области

1.1 Обзор цифровых решений для школьного питания

В настоящее время для автоматизации процессов питания и решения рутинных задач происходят попытки внедрения некоторых цифровых решений. Различные решения используют для реализации различных задач. В данный момент существует несколько примеров того, что в некоторых организациях используют программные решения для обеспечения комфортного и качественного обслуживания.

В школах Татарстана Минцифры активно внедряет системы компании ICL Services, включающие комплекс: киоск самообслуживания для самостоятельной оплаты школьных обедов и систему AiSee, обеспечивающую распознавание блюд и контроль процесса обслуживания. Данная система позволяет анализировать блюда, находящиеся на подносе, и в краткий срок производить безналичную оплату. Это значительно сокращает время обслуживания посетителей. Однако данная система направлена на автоматизацию обслуживания, но не на формирование персонализированного меню.

Технология «обедомат» предоставляет возможность получить обед с бесконтактной оплатой без участия персонала. Ученики самостоятельно оплачивают картой выбранные блюда. Однако это также не решает проблему формирования персонализированного меню для школьников, а лишь автоматизирует процесс выдачи заказов.

Система родительского контроля в Красноярске позволяет оценивать блюда, предлагаемые ученикам школы, и исключать определённый набор блюд из-за индивидуальных ограничений. Однако это не позволяет сформировать персональное меню.

Результаты сравнения приведены в таблице 1.

Как видно из результатов сравнения существующие решения направлены только на автоматизацию и контроль процессов предоставления блюд в столовых.

Таблица 1 – Результат сравнения цифровых решений

Система	Решение	Функционал	Вывод
Предзаказ через электронный дневник (Сургут)	Выбор меню родителями	Выбор одного из двух обедов	Техническая несовместимость с 4-недельным меню
«Умная столовая» (ICL Services)	Видеоаналитика + оплата	Распознавание блюд, быстрая оплата	Не формирует рекомендации, только обслуживание
Обедомат (Благовещенск)	Отложенный заказ	Заказ блюда на определённую перемену	Низкая вовлеченность родителей
Едомат	Вендинг с едой из столовой	Выдача свежих перекусов	Только холодные блюда и выпечка
Приложение родительского контроля	Контроль качества	Просмотр меню, сроков годности	Не формирует меню, только контроль

Важно отметить, что актуальность данной темы подтверждается работой Селивановой А.А. В ней автор отмечает, что «центральной проблемой составления школьного меню является необходимость одновременного учета многих нюансов: невысокой стоимости блюд, соблюдения норм СанПиНа, обеспечения разнообразия блюд»[4]. Выполнение всех трёх условий приводит к тому, что страдают вкусовые качества, что становится «основной причиной недоедания или отказа детей от питания в школе». Статистика съедаемости по возрастам. Согласно приведённым данным, только 36,4% учеников 1 – 4 классов съедают почти всю порцию, в то время как среди старшеклассников этот показатель достигает 46,5%. Одновременно доля тех, кто оставляет более половины порции, снижается с 27,2% в начальной школе до 18,0% в 10 – 11 классах.

Возрастная динамика предпочтений. Исследование выявило, что с возрастом у школьников расширяются вкусовые границы: старшеклассники чаще выбирают рассольник (21,5% против 16,6%), бефстроганов и печень, тогда как младшие школьники предпочитают более простые блюда, такие как макароны и молочные супы.

Критика нормативного подхода. Автор утверждает, что «действующие критерии планирования и контроля рационов ставят в тупик развитие системы школьного питания, так как не вполне соответствуют современному потребительскому спросу». В качестве альтернативы предлагается «исключение блюд, потребление которых составляет менее 50%» и учёт обратной связи от потребителей.

Практическая рекомендация. Эффективным методом планирования рационов автор называет «опрос самих детей после приема пищи в совокупности с оценкой количества отходов» и создание системы, позволяющей «своевременно учитывать эти данные и вносить изменения в разрабатываемое школьное меню».

1.2 Нормы СанПиН

Для разработки рекомендательной системы требуется учесть нормы санитарно–эпидемиологических требований к организации общественного питания населения. Нормы устанавливают жесткие рамки для уровня содержания калорийности, макронутриентного состава и организации хранения/приготовления блюд.

В рамках пункта 6.8 СанПиН 2.4.5.2409–08 требуется сделать рекомендательную систему для 2х разового питания. Пункт 6.9 гласит, что требуется учитывать возраст обучающихся в примерном меню. Должны быть соблюдены требования настоящих санитарных правил по массе порций блюд, их пищевой и энергетической ценности. Меню должно содержать информацию о пищевой и энергетической ценности. Производство готовых блюд осуществляется в соответствии с технологическими картами, в которых должна быть отражена рецептура и технология приготавливаемых блюд (П 6.11). Кроме того, система должна учитывать расэнергетической ценности по отдельным приёмам пищи: завтрак–25%, обед–35%. В суточном рационе питания оптимальное соотношение пищевых веществ: белков, жиров и углеводов, должна составлять 1:1:4 или в процентном отношении от калорийности, как 10–15%.

Ежедневно в рационах 2 – 6-разового питания следует включать мясо, молоко, сливочное и растительное масло, хлеб ржаной и пшеничный (с каждым приемом пищи). Рыбу, яйца, сыр, творог, кисломолочные продукты рекомендуется включать 1 раз в 2-3 дня. Завтрак требуется составлять из закуски, горячего блюда и горячего напитка, рекомендовано предоставлять овощи и фрукты.

В обед обязательно включать закуску, первое блюдо, второе блюдо (горячее из мяса, птицы или рыбы) и сладкое блюдо. В качестве закуски следует использовать салат из огурцов, помидоров, свежей или квашеной капусты, моркови, свеклы и т.п., с добавлением свежей зелени. В качестве закуски допускается использовать порционированные овощи (дополнительный гарнир). Для улучшения вкуса в салат можно добавлять свежие или сухие фрукты: яблоки, чернослив, изюм и орехи. Кроме того требуется выделять учеников в 2 возрастные группы 7 – 11 лет и 11 – 18 лет.

Рекомендуемая масса порций блюд (в граммах) представлена на рисунке 1.

Название блюд	Масса порций в граммах для обучающихся двух возрастных групп	
	с 7 до 11 лет	с 11 лет и старше
Каша, овощное, яичное, творожное, мясное блюдо	150 - 200	200 - 250
Напитки (чай, какао, сок, компот, молоко, кефир и др.)	200	200
Салат	60 - 100	100 - 150
Суп	200 - 250	250 - 300
Мясо, котлета	80 - 120	100 - 120
Гарнир	150 - 200	180 - 230
Фрукты	100	100

Рисунок 1 – Норма порций блюд

Потребность в пищевых веществах и энергии обучающихся общеобразовательных учреждений в возрасте с 7 до 11 и с 11 лет и старше представлена на рисунке 2.

Название пищевых веществ	Усредненная потребность в пищевых веществах для обучающихся двух возрастных групп	
	с 7 до 11 лет	с 11 лет и старше
Белки (г)	77	90
Жиры (г)	79	92
Углеводы (г)	335	383
Энергетическая ценность (ккал)	2350	2713
Витамин В1 (мг)	1,2	1,4
Витамин В2 (мг)	1,4	1,6
Витамин С (мг)	60	70
Витамин А (мг рет. экв)	0,7	0,9
Витамин Е (мг ток. экв)	10	12
Кальций (мг)	1100	1200
Фосфор (мг)	1650	1800
Магний (мг)	250	300
Железо (мг)	12	17
Цинк (мг)	10	14
Йод (мг)	0,1	0,12

Рисунок 2 – Потребность в пищевых веществах и энергии обучающихся общеобразовательных учреждений

Потребность в пищевых веществах и энергии обучающихся образовательных учреждений начального и среднего профессионального образования представлены на рисунке 3.

Название пищевых веществ	Потребность в пищевых веществах для обучающихся юношей и девушек	
	юноши 15 - 18 лет	девушки 15 - 18 лет
Белки (г)	98 - 113	90 - 104
в т.ч. животного происхождения	59 - 68	54 - 62
Жиры (г)	100 - 115	90 - 104
в т.ч. растительного происхождения	30 - 35	27 - 31
Углеводы (г)	425 - 489	360 - 414
Энергетическая ценность (ккал)	3000 - 3450	2600 - 2990
Витамины:		
Витамин С (мг)	70	70
Витамин А (мг рет. экв)	1,0	0,8
Витамин Е (мг ток. экв)	15	12
Тиамин (мг)	1,5	1,3
Рибофлавин (мг)	1,8	1,5
Пиридоксин (мг)	2,0	1,6
РР (мг ниац. экв)	20	17
Фолат (мкг)	200	200
Минеральные вещества:		
Кальций (мг)	1200	1200
Фосфор (мг)	1800	1800
Магний (мг)	300	300
Железо (мг)	15	18
Йод (мг)	0,13	0,13

Рисунок 3 – Потребность в пищевых веществах и энергии обучающихся образовательных учреждений

Перечень требований санитарных норм, которые требуется учесть при разработке системы указаны в таблице 2.

Таблица 2 – Перечень требований санитарных норм

Категория требований	Описание требования	Нормативный пункт
Общие санитарные нормы	Учёт норм санитарно–эпидемиологических требований к организации общественного питания (калорийность, макронутриенты, хранение, приготовление).	—
Кратность питания	Рекомендательная система разрабатывается для 2–разового питания.	Пункт 6.8
Учёт возрастных групп	Выделить две возрастные группы: – 7–11 лет – 11–18 лет	Пункт 6.9

Продолжение таблицы 2

Масса порций и ценность	Соблюдение требований к массе порций, пищевой и энергетической ценности блюд	Пункт 6.9
Информация в меню	Меню должно содержать информацию о пищевой и энергетической ценности блюд	Пункт 6.9
Технологические карты	Производство блюд — строго по технологическим картам (рецептура + технология приготовления)	Пункт 6.11
Распределение энергии по приёмам	Завтрак – 25% суточной калорийности Обед – 35% суточной калорийности	Пункт 6.8
Соотношение БЖУ	Оптимальное соотношение белки :жиры :углеводы = 1 : 1 : 4 ИЛИ в % от калорийности: белки 10–15%, жиры 25–30%, углеводы 55–60%	Пункт 6.8
Ежедневные продукты (с каждым приёмом пищи)	Мясо, молоко, сливочное масло, растительное масло, хлеб ржаной, хлеб пшеничный	Требование к рациону
Структура завтрака	Завтрак должен содержать: закуска горячее блюдо горячий напиток Рекомендовано: овощи и фрукты	Пункт 6.8
Структура обеда	Обед должен содержать: закуска первое блюдо второе блюдо (горячее из мяса, птицы или рыбы) – сладкое блюдо	Пункт 6.8

1.3 Постановка задачи

В рамках данной работы рассматривается задача автоматизации формирования комплексного школьного меню с использованием методов рекомендательных систем[5]. Актуальность задачи обусловлена необходимостью обеспечения сбалансированного питания обучающихся с учётом установленных нормативных требований, индивидуальных

особенностей и ограничений, а также повышения вариативности предлагаемых рационов.

Объектом исследования является процесс организации школьного питания. Предметом исследования выступают методы и алгоритмы формирования рекомендаций при составлении меню.

Целью работы является разработка алгоритма рекомендательной системы, обеспечивающего автоматическое формирование сбалансированного меню на основе набора исходных данных.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ существующих подходов к организации школьного питания и рекомендательных систем,
- определить требования к структуре входных данных и ограничениям,
- разработать алгоритм формирования допустимых комбинаций блюд,
- реализовать механизм оценки и ранжирования меню,
- обеспечить учёт нормативов СанПиН, индивидуальных ограничений и предпочтений пользователей,
- реализовать программную систему,
- провести экспериментальную оценку качества предложенного решения.

Входными данными системы являются:

- перечень доступных блюд с их характеристиками (калорийность, состав, категория),
- нормативные ограничения (СанПиН),
- данные о пользователях (возрастная группа, индивидуальные ограничения),
- история заказов и предпочтений.

Выходными данными системы являются:

- рекомендованные комбинации блюд для отдельных приёмов пищи;
- сформированное меню на заданный период (например, неделю);

– сопутствующие показатели качества (соответствие нормам, разнообразие, оценка предпочтений).

Ограничения задачи включают:

- соблюдение норм СанПиН по энергетической ценности и составу рациона,
- учёт индивидуальных ограничений (аллергии, запрещённые продукты),
- необходимость обеспечения разнообразия меню.

Таким образом, задача сводится к разработке алгоритма, позволяющего из множества возможных комбинаций блюд выбрать такие, которые в наибольшей степени удовлетворяют заданным ограничениям и предпочтениям пользователей.

2 Требования к системе

Для разработки наиболее оптимального решения и его адаптации под нормы СанПиН выделили требования к системе.

2.1 Требования к данным

Корректность и полнота входных данных являются ключевыми факторами, определяющими качество работы разработанной рекомендательной системы. Входные данные должны быть представлены в виде набора структурированных файлов, содержащих информацию о заказах, пользователях и характеристиках блюд.

Основной датасет должен содержать следующие поля:

- идентификатор заказа,
- наименование блюда,
- количество,
- идентификатор пользователя.

Дополнительные данные должны включать:

- информацию о пользователях (идентификатор, возраст, ограничения),
- историю заказов,
- характеристики блюд (калорийность, белки, жиры, углеводы, состав).

Требования к качеству данных.

Входные данные должны удовлетворять следующим условиям:

- отсутствуют записи с отменёнными заказами,
- значения количества являются положительными ($Quantity > 0$),
- отсутствуют пропущенные значения в ключевых полях,
- наименования блюд приведены к единому формату,
- исключены дублирующие записи.

Несоответствие данным требованиям должно устраняться на этапе предварительной обработки.

Наименования блюд должны быть нормализованы:

- приведены к единому языку представления (русский язык),

- очищены от лишних символов и вариативных написаний,
- сопоставлены с базовым словарём соответствий.

Для неизвестных или новых наименований должны применяться эвристические правила сопоставления.

Входные данные должны быть дополнены производными признаками, необходимыми для работы алгоритма.

Категориальные признаки:

- категория блюда,
- логическая категория (суп, основное блюдо, напиток и др.).

Категоризация должна выполняться на основе словаря соответствий и анализа текстовых данных.

Все блюда должны быть отнесены к одной из категорий, соответствующих структуре школьного питания:

- первое блюдо,
- горячее блюдо,
- гарнир,
- напиток,
- закуска.

Дополнительно должно быть обеспечено распределение блюд по слотам меню:

- первое блюдо,
- основное блюдо,
- гарнир,
- напиток.

Для каждого блюда должны быть определены показатели пищевой ценности:

- калорийность,
- содержание белков,
- содержание жиров,

- содержание углеводов,
- масса порции.

Допускается использование внешних источников данных (например, сборников рецептов) при соблюдении следующих условий:

- данные должны быть извлечены в текстовом виде,
- должна быть обеспечена фильтрация шумовых записей,
- должны быть выделены корректные наименования блюд,
- исключены дубликаты и некорректные записи.

Результирующий набор данных должен быть приведён к формату, совместимому с основной системой.

Все входные данные должны быть согласованы между собой:

- идентификаторы пользователей должны совпадать во всех файлах,
- наименования блюд должны быть унифицированы,
- характеристики блюд должны соответствовать используемым наименованиям.

Нарушение согласованности должно устраняться до этапа формирования рекомендаций.

2.2 Требования к алгоритму

Алгоритм должен базироваться на принципах гибридной рекомендации. Включать в себя: знание–ориентированный подход (физиологические нормы, правила составления рациона); ограничения на основе правил (rule-based reasoning) для обеспечения периодичности, разнообразия и структурной целостности приемов пищи. Программа должна быть разделена на несколько модулей, каждый отвечающий за формирование определённого правила.

Модуль структурного контроля приемов пищи: Для завтрака формирует обязательное наличие трех компонентов: закуска, горячее блюдо, горячий напиток. Для обеда – обязательное наличие четырех компонентов: салат, первое блюдо, второе блюдо, сладкое блюдо.

Модуль ограничений периодичности: реализует механизм, запрещающий использование одного и того же блюда или продукта из одной группы чаще,

чем 1 раз в 3 дня. Обеспечивает контроль обязательных ежедневных продуктовых групп (мясо, молоко, масло, хлеб) в дневном рационе.

Модуль калорийного нормирования: Распределение суточной калорийности по приемам пищи должно соответствовать нормам: завтрак – 25%, обед – 35%. Использует возрастные нормы суточного потребления калорий в качестве базового ограничения.

3 Разработка рекомендательной системы

3.1 Описание алгоритма

Реализация алгоритма рекомендательной системы должна учитывать множество ограничений и вычислять промежуточные параметры для предоставления результата с соблюдением санитарных норм. Для достижения поставленной задачи на этапе инициализации программа должна определить возрастную группу студента, на основании этого получить данные из справочника по конкретной возрастной группе. Затем получить из справочника структуру приема и список доступных блюд для каждой категории блюд. На основании этих данных рассчитать требуемую норму нутриентов с учетом периодичности, калорийности и ограничений пользователя.

В зависимости от возраста ученику будет присваиваться 1 или 2 возрастная группа на этапе зачисления и обновляться в том, случае если ученик попадает в старшую категорию.

Для ученика s из истории покупок Hs извлекаются записи за последние 3 месяца. Число недель в анализируемом окне:

$$W = \max(5(\max(\text{date}) - \min(\text{date})).\text{days}, 1), \quad (1)$$

где: $\max(\text{date})$ – дата начала анализируемого периода;

$\min(\text{date})$ – дата окончания анализируемого периода.

Частота потребления блюда d в категории c (раз в неделю):

$$fs(d, c) = \text{count}(d, c, HsT) / Wfs, \quad (2)$$

где: d – выбранное блюдо;

c – категория выбранного блюда.

Ранжирование и TOP-K рекомендации:

Для каждой пары (s, c) блюда упорядочиваются по убыванию. При равной частоте используется стабильная сортировка по порядку в справочнике.

Для учеников без истории покупок возвращаются первые k блюд из $A(s, c)$ в порядке справочника.

3.2 Выбор метода фильтрации

Для наиболее эффективной работы рекомендательной системы разберем существующие подходы фильтрации.

Один из методов является коллаборативная фильтрация. Это метод в рекомендательных системах использующий оценки группы пользователей для прогнозирования неизвестных предпочтений другого пользователя. Основная идея метода заключается в выявлении оценок пользователя, для которого нужно спрогнозировать рекомендации. Затем система анализирует, что выбирают пользователи с похожими запросами и на основании этих результатов предлагает рекомендации.

Контентная фильтрация – метод, который проводит анализ предпочтений конкретного пользователя. Он основывается на характеристиках тех объектов, которые уже когда-либо выбирал пользователь.

Следующий подход сочетает в себе оба вышеупомянутых метода и предоставляет рекомендации, основываясь на них.

Методы ближайших соседей (k -NN) являются одним из наиболее распространённых подходов в коллаборативной фильтрации. Основная идея метода заключается в поиске объектов, наиболее похожих друг на друга, и использовании их поведения для формирования рекомендаций. Основные преимущества метода – это простота реализации, интерпретируемость. Среди недостатков данного метода можно отметить плохое масштабирование на большом числе объектов. Также он не учитывает сложные зависимости и чувствителен к разряженности данных.

Графовые методы рассматривают задачу рекомендаций как задачу анализа графа. Для построения рекомендаций могут использоваться:

- поиск соседей в графе,
- случайные блуждания (random walk),
- алгоритмы распространения влияния.

На практике в данной работе используется упрощённый вариант – анализ матрицы совместной встречаемости, полученной из ассоциативных правил.

Преимущества графового подхода:

- хорошо моделирует связи между блюдами,
- устойчив к разреженности данных,
- естественно учитывает структуру меню.

Недостатки:

- сложность реализации,
- необходимость настройки параметров,
- вычислительная сложность.

Основные различия методов представлены в таблице 3.

Таблица 3 – Сравнение методов фильтрации

Тип	Принцип работы	Пример применения
Коллаборативная фильтрация	Рекомендует то, что понравилось похожим пользователям	«Вам также понравится...» в онлайн-кинотеатрах
Контентная фильтрация	Рекомендует блюда, похожие на те, что пользователь уже заказывал	Учёт категории блюда, ингредиентов, калорийности
Гибридные модели	Комбинируют оба подхода	LightFM, Deep Learning рекомендации
K-NN	Находит наиболее похожие объекты на основе метрик сходства и рекомендует на основе их выбора	Рекомендация блюд, которые часто выбираются вместе
Графовые методы	Представляют данные в виде графа связей между объектами и используют её для рекомендаций	Формирование комплексного меню на основе часто сочетаемых блюд

Использование коллаборативного подхода не позволит учитывать требования СанПиН, возрастные нормы и пищевые ограничения, так как рекомендации предоставляются на основании заказов группы учеников. При этом система не может проверить, соответствует ли это нормам или нет. Следовательно, этот метод не подходит.

При использовании контентной фильтрации система не будет учитывать индивидуальные предпочтения учеников. Следовательно, этот подход не подойдёт для реализации данного метода.

Целесообразно использовать гибридный подход, так как он объединяет сильные стороны обоих методов и компенсирует их недостатки[6]. Нормативные и персональные ограничения возможно учесть с помощью методов контентной фильтрации путём настройки фильтров по атрибутам и составу блюд. Формирование предпочтений можно реализовать посредством коллаборативной фильтрации на следующем этапе. На основании получившихся данных можно формировать персонализированное меню.

3.3 Механизм формирования меню

Алгоритм генерации недельного меню должен реализовывать следующий итеративный процесс: Определение целевой калорийности для каждого приема пищи на основе профиля пользователя. Формирование корзины доступных блюд с учетом типа приема пищи, истории потребления за последние 3 дня, диетических ограничений пользователя. Компоновка приема пищи путем последовательного выбора блюд. Соответствие структурным требованиям и суммарной калорийности/нутриентного баланса.

4 Реализация системы

4.1 Подготовка данных

На первом этапе выполняется загрузка исходного CSV-файла с данными о продажах блюд. Входной датасет содержит информацию о заказах: идентификатор заказа, название блюда, количество, признак отмены заказа, кем был заказан.

Очистка данных является критически важным этапом, так как исходные данные содержат шум и неиспользуемые записи. Для использования выбранного датасета требуется реализовать следующие процедуры:

- удаление отменённых заказов,
- фильтрация записей с некорректным количеством ($\text{Quantity} \leq 0$),
- удаление строк с пропущенными значениями,
- нормализация названий блюд.

Для последующего анализа и построения рекомендаций были сформированы дополнительные признаки:

- временные характеристики: день недели, месяц и час заказа,
- категориальные признаки: категория блюда (из исходного датасета), логическая категория (soup, main, drink и т.д.).

Категоризация выполняется с использованием словаря соответствий и ключевых слов в названии блюда. Это позволяет структурировать неформализованные текстовые данные.

Для семантического преобразования блюд используется базовый словарь соответствий, для неизвестных блюд применяются эвристические правила. Это позволяет получить интерпретируемый набор данных для школьной среды.

Введена дополнительная классификация для категоризации по нормам школьного питания:

- первое,
- горячее,
- гарнир,
- напиток,

- закуска.

Также реализовано распределение блюд по слотам подноса:

- перовое,
- основное блюдо,
- гарнир,
- напиток.

Это необходимо для дальнейшей генерации сбалансированного меню.

Поскольку в исходном датасете отсутствуют данные о пищевой ценности, была создана синтетическая база нутриентов на основании существующих источников. Для каждой категории блюда задаются типовые значения:

- калорийность,
- белки,
- жиры,
- углеводы,
- масса порции.

Для формирования базы блюд подготовим данные из сборника рецептов блюд[7]. Для этого был написан парсер, который позволил считать PDF–документ, содержащий информацию о рецептуре блюд.

Процесс обработки PDF включал несколько этапов:

- извлечение полного текста документа,
- предварительная очистка текста,
- выделение кандидатов в названия блюд,
- дополнительная фильтрация и устранение шума.

Для реализации использовалась библиотека PyMuPDF (fitz), позволяющая извлекать текст из PDF–файлов с сохранением порядка чтения. На первом этапе весь текст документа извлекался постранично: каждая страница обрабатывалась отдельно. Текст объединялся в единый строковый массив и сохранялся в файл full_text.txt. После извлечения текста выполнялась его нормализация: удаление

лишних пробелов, замена неразрывных пробелов, приведение строк к единому формату. Извлечение текста происходит с помощью функции `extract_full_text`.

```
def extract_full_text(pdf_path: Path) -> str:
    try:
        import fitz
    except ImportError as e:
        doc = fitz.open(pdf_path)
        parts: list[str] = []
        for i in range(doc.page_count):
            page = doc.load_page(i)
            parts.append(page.get_text("text"))
        doc.close()
    return "\n\n".join(parts)
```

Текст разбивается на отдельные строки и абзацы (по пустым строкам). Это необходимо для дальнейшего анализа структуры документа. Для определения строк, содержащих названия блюд, используется набор эвристических правил. Поиск выполняется по абзацам документа. Каждая новая строка проверяется на наличие названия блюда.

Дополнительно анализируются все строки документа: каждая строка проверяется на соответствие критериям названия блюда, подходящие строки добавляются в список кандидатов. Функция `_maybe_dish_title()` реализует ключевую логику фильтрации.

```
def _maybe_dish_title(line: str) -> bool:
    if len(line) > 160 or len(line) < 4:
        return False
    if RE_SKIP.search(line):
        return False
    digit_ratio = sum(c.isdigit() for c in line) /
max(len(line), 1)
    if digit_ratio > 0.35:
        return False
        if line.count(",") >= 4 and
re.search(r"\d+\s*(г|кг|мл)\b", line, re.I):
        return False
```

```

if RE_LEADING_CODE.match(line):
    return True
if 8 <= len(line) <= 120 and line[0].isalpha():
    if line.count(".") <= 1:
        return True
return False

```

Строка считается названием блюда, если: длина строки от 4 до 160 символов, строка начинается с буквы, содержит преимущественно текст, а не цифры. Отбрасываются строки, содержащие: только цифры или веса; Слова: «состав» «ингредиенты» «технологическая карта» «выход» большое количество чисел.

Дополнительные признаки:

- строка может начинаться с номера рецепта (например, “12. Борщ”),
- допускаются короткие текстовые заголовки без точки.

Дополнительно применяется метод выделения строк, написанных полностью заглавными буквами:

- такие строки часто являются заголовками блюд,
- фильтруются строки, содержащие только символы верхнего регистра,
- исключаются строки с малым количеством букв.

Результат сохраняется в файл `lines_all_caps.txt`.

После применения всех фильтров:

- дубликаты удаляются,
- список сортируется по алфавиту,
- каждому блюду присваивается порядковый номер.

Итоговый список сохраняется в файл: `dishes_candidates.txt`

Функция `_letters_only` Функция извлекает из строки только буквенные символы.

```

def _letters_only(s: str) -> str:
    return "".join(c for c in s if c.isalpha())

```

Функция `collect_dish_candidates` формирует список кандидатов в названия блюд.

```

def collect_dish_candidates(full_text: str) -> list[str]:
    candidates: set[str] = set()
    paragraphs = re.split(r"\n\s*\n+", full_text)
    for para in paragraphs:
        lines = [_normalize_line(x) for x in para.split("\n")
if _normalize_line(x)]
        if not lines:
            continue
        first = lines[0]
        if _maybe_dish_title(first):
            candidates.add(first)
    for line in full_text.split("\n"):
        line = _normalize_line(line)
        if not line or len(line) < 5:
            continue
        if _maybe_dish_title(line):
            candidates.add(line)
    return sorted(candidates, key=lambda x: x.lower())

Функция main():
def main() -> None:
    parser = argparse.ArgumentParser(description="Извлечь
текст и кандидаты в названия блюд из PDF.")
    parser.add_argument(
        "pdf",
        nargs="?",
        default=None,
        help="Путь к PDF (по умолчанию:
СБОРНИК РЕЦЕПТУР БЛЮД 2.4.pdf в этой папке)",
    )
    parser.add_argument(
        "--min-caps-letters",
        type=int,
        default=4,
        metavar="N",
        help="Минимум букв в строке для lines_all_caps.txt
(по умолчанию 4)",

```

```

    )
    args = parser.parse_args()
    pdf_path = Path(args.pdf) if args.pdf else
_default_pdf_path()
    if not pdf_path.is_file():
        print(f"Файл не найден: {pdf_path}", file=sys.stderr)
        print("Положите PDF в папку pdf_recipes_parser или
укажите полный путь.", file=sys.stderr)
        sys.exit(1)
    out_dir = _script_dir() / "out"
    out_dir.mkdir(parents=True, exist_ok=True)
    text = extract_full_text(pdf_path)
    (out_dir / "full_text.txt").write_text(text,
encoding="utf-8")
    dishes = collect_dish_candidates(text)
    lines_out = [f"{i + 1}. {name}" for i, name in
enumerate(dishes)]
    (out_dir / "dishes_candidates.txt").write_text(
        "\n".join(lines_out) + "\n",
        encoding="utf-8",
    )
    caps_lines = collect_all_caps_lines(text,
min_letters=args.min_caps_letters)
    caps_out = [f"{i + 1}. {name}" for i, name in
enumerate(caps_lines)]
    (out_dir / "lines_all_caps.txt").write_text(
        "\n".join(caps_out) + "\n",
        encoding="utf-8",
    )
    meta = (
        f"Источник: {pdf_path}\n"
        f"Символов в full_text: {len(text)}\n"
        f"Кандидатов в названия: {len(dishes)}\n"
        f"Строк только CAPS (уник., порядок как в PDF, min
букв={args.min_caps_letters}): {len(caps_lines)}\n"
    )

```

```

        (out_dir / "extraction_meta.txt").write_text(meta,
encoding="utf-8")

    print(meta)
    print(f"Записано: {out_dir / 'full_text.txt'}")
    print(f"Записано: {out_dir / 'dishes_candidates.txt'}")
    print(f"Записано: {out_dir / 'lines_all_caps.txt'}")
if __name__ == "__main__":
    main()

```

Используемые входные файлы:

- dishes_nutrition.csv – справочник блюд, содержащий поля: dish_name (название блюда), category(категория: "Каша", "Блюдо из мяса", "Гарнир" и др.), calories(калорийность), protein (белки), fat (жиры), carbs (углеводы),
- students.csv – данные учеников, содержащие поля: student_id (идентификатор), age (возраст) restrictions (медицинские ограничения: "Безмолочная диета", "Безглютеновая диета", "Рыбная аллергия", "Диабет"),
- school_canteen_full_dataset.csv – история покупок, содержащая поля: date (дата), student_id (идентификатор ученика), dish_name (название блюда), category (категория), price (цена)[8].

Эти данные используются для составления рейтинга блюд и проверки жестких ограничений учеников.

4.2 Архитектура системы

Разработанная система формирования рекомендаций комплексного школьного меню реализована в виде модульного программного решения на языке Python, обеспечивающего последовательную обработку входных данных, частотный анализ предпочтений, фильтрацию по медицинским ограничениям и генерацию персонализированного недельного меню с соблюдением санитарных норм.

Архитектура системы построена по конвейерному принципу (pipeline) и включает следующие основные компоненты: модуль загрузки и предобработки данных, модуль временного разбиения, модуль медицинской фильтрации,

модуль частотного профилирования предпочтений, модуль оценки качества, модуль санитарного нормирования, модуль планирования меню и модуль генерации выходных данных.

Модуль загрузки и предобработки данных отвечает за чтение входной информации из CSV–файлов и первичную валидацию данных. Поддерживается загрузка трёх основных источников:

В процессе загрузки осуществляется проверка целостности данных: отсутствие пропущенных значений в ключевых полях, соответствие категорий блюд допустимому перечню, корректность формата дат. Результатом работы модуля является формирование внутренних структур данных в виде `pandas.DataFrame`, обеспечивающих эффективный доступ и фильтрацию.

Модуль временного разбиения (`src/split.py`). На этапе предобработки выполняется разделение исторических данных на три непересекающиеся выборки по временному признаку. Такой подход имитирует реальный сценарий работы системы, когда обучение производится на прошлых данных, а проверка – на будущих.

Все уникальные даты в истории покупок сортируются в хронологическом порядке и распределяются в следующей пропорции:

- обучающая выборка (`train`) – первые 60% дней, используется для расчёта частот потребления,
- тестовая выборка (`test`) – следующие 30% дней, используется для оценки точности рекомендаций,
- валидационная выборка (`val`) – последние 10% дней, используется для финальной верификации.

При объёме исходных данных в 130 дней формируется: 78 дней обучения ($\approx 78\,000$ записей), 39 дней тестирования ($\approx 39\,000$ записей), 13 дней валидации ($\approx 13\,000$ записей). Разбиение выполняется функцией `temporal_split(purchases, train_ratio=0.6, test_ratio=0.3, val_ratio=0.1)`.

Модуль медицинской фильтрации (`src/restrictions.py`) отвечает за исключение блюд, противопоказанных ученику по медицинским показаниям.

Фильтрация реализована через правила на основе ключевых слов в названии блюда и его категории.

Для каждого типа ограничения определены детерминированные правила (Таблица 4):

Таблица 4 – Детерминированные правила

Ограничение	Принцип фильтрации	Исключения (разрешённые блюда)
Безмолочная диета	Исключаются блюда, содержащие ключевые слова: молоко, сливки, творог, сыр, йогурт, кефир, ряженка	Омлет без сыра, яйцо варёное
Безглютеновая диета	Исключаются блюда с ключевыми словами: пшеница, рожь, ячмень, овёс, макароны, булочка, кекс, перловка	Гречка, рис, булгур, безглютеновые макароны
Диабет	Исключаются блюда с ключевыми словами: сахар, мёд, сладкие каши, сладкое мучное	Стевия, фруктоза, сорбит

Функция `allowed(dish_name, category, restrictions)` возвращает булево значение, определяющее, разрешено ли блюдо для ученика с заданными ограничениями. Множество разрешённых блюд для ученика `ss` в категории `c` формально описывается как:

$$A(s,c)=\{d \mid d \in \text{категории } c \wedge \text{allowed}(d,c,R_s)=\text{True}\}, \quad (3)$$

где: `d` – название блюда;

`c` – категория блюда.

Модуль частотного профилирования предпочтений (`src/preferences.py`) реализует ядро рекомендательной логики – определение того, какие блюда ученик предпочитает, на основе исторических данных об успешных покупках.

Расчёт частоты потребления. Для каждого ученика s из обучающей выборки извлекаются записи за последние 3 месяца (параметр $PREFERENCE_WINDOW_MONTHS=$). Число недель в анализируемом окне вычисляется как:

$$W=\max(7(\max(\text{date})-\min(\text{date})).\text{days},1), \quad (4)$$

где: $\max(\text{date})$ – дата начала анализируемого периода;

$\min(\text{date})$ – дата окончания анализируемого периода.

Частота потребления блюда d категории c (раз в неделю) определяется отношением числа покупок этого блюда к количеству недель в окне:

$$fs(d,c)=\text{count}(d,c,HsT)Wfs, \quad (5)$$

где: d – название блюда;

c – категория блюда.

Ранжирование и TOP-K рекомендации. Для каждой пары «ученик–категория» блюда из множества $A(s,c)$ упорядочиваются по убыванию $fs(d,c)$. При равной частоте используется стабильная сортировка по порядку в справочнике. Функция $\text{TopK}(s,c,k=3)$ возвращает первые k блюд. Для учеников без истории покупок возвращаются первые k блюд из $A(s,c)$ в порядке справочника.

Определение предпочтительных категорий выполняется по формуле 6. Для завтрака система определяет предпочтение между категориями «Каша» и «Блюдо из яиц и творога»:

$$\text{pref_main}(s) = \operatorname{argmax}_{c \in \{ \text{"Каша"}, \text{"Блюдо из яиц и творога"} \}} \quad (6)$$

где: TopK – топ блюд,

d – блюдо,

s – ученик,

c – категория блюда.

Модуль оценки качества рекомендаций (src/evaluator.py) предназначен для измерения точности рекомендаций на тестовой и валидационной выборках. Основной метрикой является HitRate@K (при K=3).

4.2 Реализация алгоритма

Разработанная система рекомендаций реализована в виде программного модуля на языке Python и предназначена для формирования персонализированного комплексного меню школьников с учетом нормативных и индивидуальных ограничений.

В процессе разработки применялись стандартные библиотеки Python, включая collections, itertools, glob и os, а также встроенные средства обработки строк и файлов.

Алгоритм рекомендательной системы реализован без использования специализированных библиотек машинного обучения, что позволило обеспечить прозрачность логики принятия решений и гибкость настройки. Реализация включает модули загрузки и предобработки данных, формирования допустимых комбинаций блюд, а также вычисления оценочной функции с учётом норм СанПиН, индивидуальных ограничений пользователей и истории заказов.

Система включает несколько последовательных этапов обработки данных. На первом этапе выполняется загрузка и парсинг исходных данных, включающих информацию о блюдах, составе ингредиентов, доступности продуктов на складе, нормах СанПиН, характеристиках учеников, истории заказов и пользовательских рейтингах. Все данные приводятся к единому структурированному виду для последующей обработки.

На втором этапе осуществляется формирование всех возможных комбинаций блюд. Для завтрака формируются комбинации из трех элементов (основное блюдо, напиток и мучное изделие), для обеда — из пяти элементов (первое блюдо, гарнир, горячее, напиток и мучное изделие). Перебор

комбинаций реализуется с использованием декартового произведения множеств блюд соответствующих категорий.

`__init__(self, dishes)` выполняет инициализацию модели предпочтений.

Функция принимает справочник блюд в виде DataFrame и сохраняет его в свойстве `self.dishes`. Затем все блюда группируются по категориям и строится словарь `self._category_dishes`, где ключом является название категории (например, "Каша", "Блюдо из мяса"), а значением – список названий блюд, входящих в эту категорию. Это делает доступ ко всем блюдам нужной категории значительно быстрее при последующей фильтрации и ранжировании.

`_window_filter(self, history, ref_date)` фильтрует историю покупок за последние N месяцев.

Функция вычисляет начальную дату как `ref_date` минус количество месяцев, заданное в конфигурации (по умолчанию 3 месяца). Затем в переданной истории отфильтровываются записи, у которых дата покупки больше или равна начальной дате. В результате возвращается DataFrame, содержащий только покупки за последние 3 месяца.

`compute_frequencies(self, student_id, history, ref_date)` выполняет расчёт частоты каждого блюда для конкретного ученика (в разгах в неделю).

Функция сначала отфильтровывает историю по идентификатору ученика. Затем выполняет фильтрацию через `_window_filter`, чтобы оставить только записи за последние 3 месяца. Далее она определяет минимальную и максимальную даты в отфильтрованных данных и вычисляет количество недель в этом промежутке (если разница меньше недели, устанавливается значение 1). После этого функция группирует покупки по паре "категория — название блюда", подсчитывает количество покупок для каждой группы и делит это количество на количество недель. Результат возвращается в виде вложенного словаря: сначала ключ – категория, затем ключ – название блюда, значение – частота в неделю.

`top_dishes(self, student_id, category, history, restriction, ref_date, k)` получает TOP-K любимых блюд ученика в указанной категории.

Функция сначала вызывает `compute_frequencies`, чтобы получить частоты всех блюд ученика. Затем она получает из `_category_dishes` полный список всех блюд запрошенной категории. Даже если у ученика нет истории, список всех блюд сохраняется для дальнейшей фильтрации. После этого функция вызывает внешнюю функцию `filter_dish_names`, которая исключает из списка блюда, запрещённые из-за аллергии или диабета.

Если после фильтрации список разрешённых блюд пуст, функция возвращает пустой список. Если у ученика нет истории (словарь частот для этой категории пуст), то функция возвращает первые `k` блюд из отфильтрованного списка. Если история есть, то функция сортирует разрешённые блюда по убыванию частоты (блюда с большей частотой идут первыми), а при равной частоте сохраняет исходный порядок из списка разрешённых блюд. Затем функция возвращает первые `k` элементов из отсортированного списка.

`high_probability_dishes(self, student_id, category, history, restriction)` позволяет получить список блюд, которые ученик берёт с высокой вероятностью.

Функция вызывает `compute_frequencies` для получения частот всех блюд ученика. Затем она извлекает словарь частот для запрошенной категории и отбирает по нему только те блюда, у которых частота больше или равна пороговому значению `HIGH_FREQ_THRESHOLD_PER_WEEK` (по умолчанию 1.0). Полученный список названий блюд дополнительно фильтруется через `filter_dish_names` для учёта ограничений по здоровью. Результатом является список выбранных блюд.

`preferred_breakfast_main(self, student_id, history, restriction)` определяет предпочтения учеников на завтрак.

Функция работает со списком из двух категорий: "Каша" и "Блюдо из яиц и творога". Для каждой категории она вызывает `top_dishes` с `k=5`, чтобы получить пять самых популярных блюд в этой категории. Затем через `compute_frequencies` она получает частоты и суммирует частоты этих пяти

блюд — получается некий "вес" категории, отражающий общую популярность этой группы блюд у ученика.

После подсчёта весов для обеих категорий функция сравнивает их. Если обе суммы равны нулю (то есть у ученика нет истории покупок), функция возвращает "Каша" как значение по умолчанию. В противном случае возвращается название категории с большей суммой частот.

`preferred_lunch_protein(self, student_id, history, restriction)` определяет предпочтения ученика на обед — что он чаще выбирает: мясные или рыбные блюда.

Функция работает со списком из двух категорий: "Блюдо из мяса" и "Блюдо из рыбы". Для каждой категории она сначала получает все разрешённые блюда через `filter_dish_names` (с учётом аллергии и диабета). Затем через `compute_frequencies` она получает частоты и суммирует частоты всех разрешённых блюд этой категории — так вычисляется общий "вес" категории.

Если после суммирования обе суммы равны нулю, функция возвращает "Блюдо из мяса" как значение по умолчанию. В противном случае возвращается название категории с большей суммой частот.

`predict_top_k(self, student_id, category, history, restriction, k)` предсказывает ТОП-К блюд для оценки качества модели на тестовых данных.

Алгоритм включает этапы загрузки и предобработки данных, формирования допустимых комбинаций блюд, их оценки с учётом заданных ограничений и предпочтений пользователей, а также выбора наилучших вариантов для формирования итогового меню.

Схема алгоритма представлена на рисунке 4.

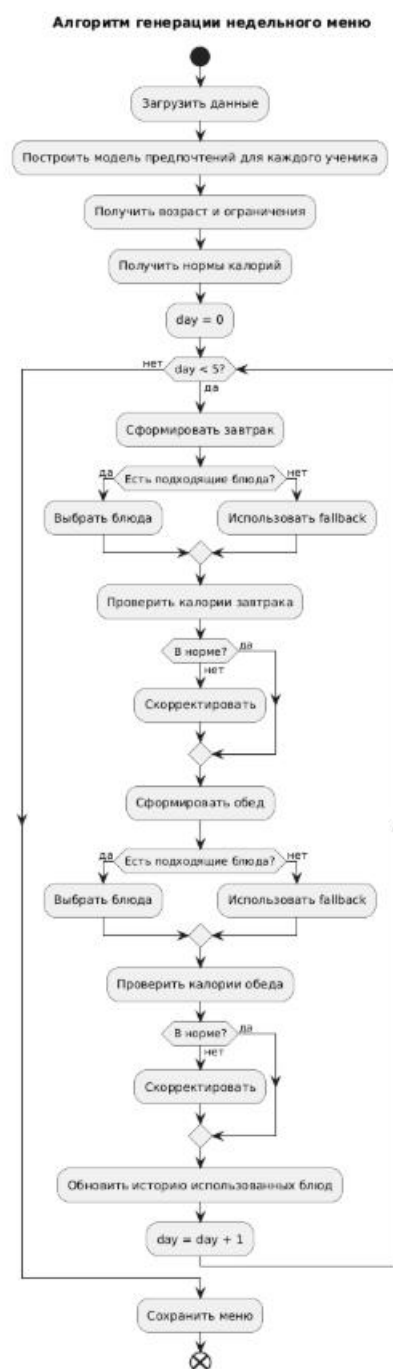


Рисунок 4 – Схема алгоритма

Представление алгоритма в виде онтологической модели на рисунке 5. Модель описывает структуру школьного питания через систему классов, подклассов, объектных свойств и признаков-ограничений, что позволяет автоматически формировать непротиворечивое меню с учётом санитарных норм (СанПиН), возрастных групп и индивидуальных ограничений здоровья

Классы и подклассы. Приём пищи является абстрактным классом, его подклассы: Завтрак (3 слота) и Обед (5 слотов). Слот – это позиция внутри

приёма пищи, например первое, мясо или рыба, гарнир. Блюдо — абстрактный класс, его подклассы: Каша, Яйца или творог, Мясо, Рыба, Первое, Гарнир, Мучное, Горячий напиток, Напиток. Возрастная группа включает подклассы: 7–11 лет и 12+ лет. Меню делится на подклассы: Меню на день, Меню на неделю, Меню ограниченное (с учётом аллергии и диабета).

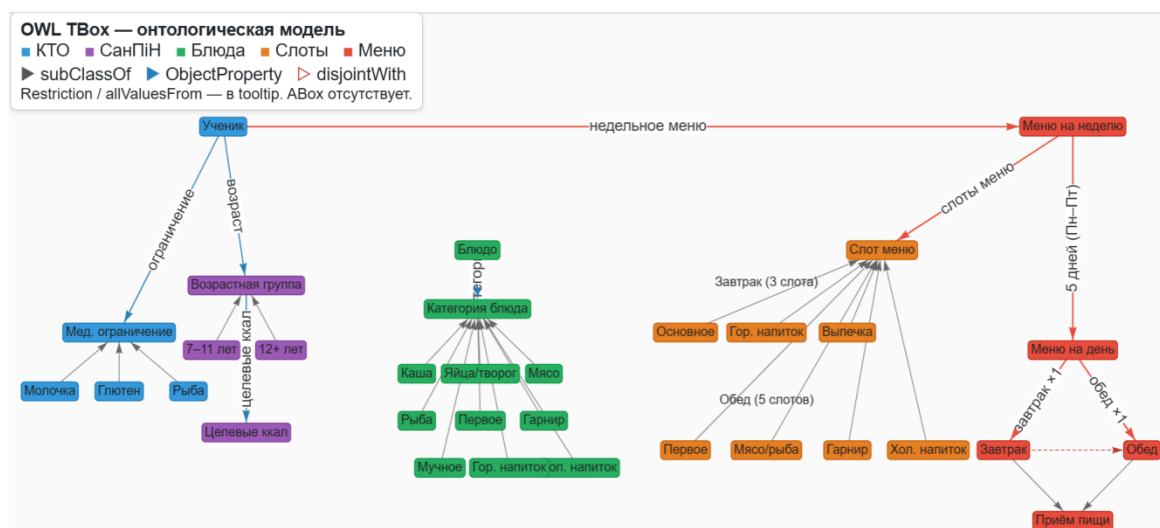


Рисунок 5 – Онтологическая модель алгоритма

5 Эксперимент и оценка

Для наиболее удобного отображения полученного меню был разработан веб интерфейс, предоставляющий всю необходимую информацию о требуемом количестве блюд. Рекомендательная система предоставляет API для получения недельного меню. Взаимодействие между клиентской частью и сервером обеспечивается посредством HTTP-запросов. Обмен данными производится в формате JSON, что обеспечивает универсальность и лёгкость интеграции с любыми веб-приложениями. Метаданные содержат информацию о названии блюда, его категории, дне приёма, возрастной группы. Структура json:

```
{
  "meta": {
    "source": "scenario_canteen",
    "mode": "with_orders",
    "student_count": 2,
    "days_per_week": 1
  },
  "students": {
    "Алиса": {
      "age": 9,
      "age_group": "младшие",
      "days": [
        {
          "day": "Понедельник",
          "age_group": "младшие",
          "breakfast": {
            "dishes": [
              { "role": "main", "label": "основное", "name":
"Омлет с овощами" },
              { "role": "drink", "label": "напиток", "name":
"Компот" },
              { "role": "bakery", "label": "мучное", "name":
"Рисовая лепёшка" }
            ],
            "nutrition": { "cal": 460, "p": 17, "f": 20, "c":
59 },
            "score": 0.6008
          },
          "lunch": {
            "dishes": [
              { "role": "first", "label": "первое", "name":
"Борщ" },
              { "role": "garnish", "label": "гарнир", "name":
"Гречка гарнир" },
              { "role": "main", "label": "горячее", "name":
"Овощные котлеты" },
```

112 } ,

ограничений (рис. 6 – 7).

Рисунок 6 – Сформированное меню для 1 – 3 дня недели

Четверг		Пятница		Суббота	
ЗАВТРАК		ЗАВТРАК		ЗАВТРАК	
Блюдо	Каша вязкая молочная кукурузная	Блюдо	Каша жидкая молочная кукурузная	Блюдо	Творожно-пшениная запеканка
Нетто	167 г	Нетто	171 г	Нетто	145 г
Блюдо	Чай с молоком и сахаром	Блюдо	Чай с сахаром	Блюдо	Чай с лимоном и сахаром
Нетто	188 г	Нетто	170 г	Нетто	166 г
Блюдо	Кекс Столичный	Блюдо	Ватрушка творожная	Блюдо	Булочка школьная
Нетто	84 г	Нетто	81 г	Нетто	85 г
ОБЕД		ОБЕД		ОБЕД	
Блюдо	Салат из моркови и яблок	Блюдо	Салат из белокочанной капусты с морковью и яблоками	Блюдо	Салат из белокочанной капусты с помидорами и огурцами
Нетто	103 г	Нетто	101 г	Нетто	100 г
Блюдо	Рассольник домашний	Блюдо	Суп сливочный с рыбой (горбуша)	Блюдо	Суп крестьянский с крупой (рисовая)
Нетто	203 г	Нетто	184 г	Нетто	197 г
Блюдо	Котлеты из курицы	Блюдо	Биточек из курицы	Блюдо	Котлета рыбная (треска)
Нетто	93 г	Нетто	96 г	Нетто	98 г
Блюдо	Капуста белокочанная тушеная	Блюдо	Картофельное пюре	Блюдо	Рагу из овощей
Нетто	148 г	Нетто	137 г	Нетто	142 г
Блюдо	Напиток из шиповника	Блюдо	Компот из кураги	Блюдо	Напиток из шиповника
Нетто	180 г	Нетто	180 г	Нетто	170 г
Сумма за день: Ккал=1299.7 · Б=44.9 · Ж=44.6 · У=176.8 г		Сумма за день: Ккал=1578.0 · Б=60.1 · Ж=60.1 · У=184.0 г		Сумма за день: Ккал=1619.2 · Б=64.7 · Ж=63.0 · У=181.5 г	

Рисунок 7 – Сформированное меню для 4 –6 дня недели

На основе полученных данных меню для ученика младшей группы можно сделать следующие выводы: набор блюд меняется частично, что формально соответствует норме СанПин, но не удовлетворяет требование полностью. Средняя калорийность на день составляет $(1532.9 + 1374.6 + 1218.6 + 1299.7 + 1578 + 1619.2) / 6 = 1437,2$ ккал/день. С учетом распределения калорий (25% – завтрак и 35 % – обед) при рекомендованной суточной норме 2350 ккал, на завтрак и обед должно приходиться $2350 \cdot 0.6 = 1410$. Фактический результат больше, следовательно требование выполняется. Фактический показатель белков в среднем 52–64 г/день. Показатель жиров: в среднем 43–63 г/день. Показатель углеводов: в среднем 145–184 г/день. Среднее соотношение БЖУ в неделю составляет 1.09 : 1.00 : 3.10, удовлетворяет норму СанПин с небольшим отклонением.

5.1 Сравнение со стандартным подходом

В рамках исследования было проведено сравнение двух подходов к формированию меню в школьной столовой.

Произвольное общее меню:
(файл scenario_canteen/arbitrary_week.txt) – единый недельный комплексный рацион для всех учащихся.

Персонализированное меню, формируемое функцией build_weekly_for_student() – индивидуальный подбор рациона для каждого ученика.

Для этого в первом случае меню генерировалось функцией deviation_score, которая отвечает за соответствие нормам сан пина.

Во втором случае генерация учитывала заказы учеников, затем формировала меню с учетом популярных блюд.

Сравнение проводилось по следующим ключевым критериям(таблица 4):

Таблица 4 – Сравнительный анализ критериев используемых подходов

Критерий	Произвольное меню	Персонализированное меню
Соответствие СанПиН	+	+
Учёт истории заказов	–	+
Ограничения (аллергии, религия)	Учитывается в случае, если предоставлены особые документы. Контролируется вручную	Наличие ограничений указывается в исходных данных
Разнообразие	+	+
Применимость	Массовая	Массовая, из расчета наиболее популярных блюд среди всех учеников

Важно отметить, что «произвольное меню» не является полностью случайным – оно соответствует типичной практике столовых: соответствует СанПиН, но не адаптировано под индивидуальные особенности.

5.2 Методика и результаты сравнения текущего алгоритма и ML – подходов

Для оценки эффективности разработанного алгоритма было проведено экспериментальное сравнение с двумя современными методами машинного обучения, предназначенными для задач ранжирования: CatBoostRanker и

XGBRanker. Выбор данных методов обусловлен их высокой эффективностью в задачах рекомендации с использованием градиентного бустинга, а также наличием встроенной поддержки оптимизации метрик ранжирования, таких как NDCG (Normalized Discounted Cumulative Gain).

Для каждого ученика и каждой категории блюд (завтрак, обед, гарнир, напитки и т.д.) требовалось предсказать наиболее вероятное блюдо, которое ученик выберет в следующий раз. Задача формулировалась как ранжирование: для каждого ученика и категории необходимо было выдать упорядоченный список из наиболее релевантных блюд.

Для обеспечения репрезентативности оценки все данные были разделены на три выборки:

- обучающая выборка (60%) – использовалась для настройки параметров моделей,
- валидационная выборка (10%) – применялась для контроля переобучения и подбора гиперпараметров,
- тестовая выборка (30%) – служила для финальной оценки качества моделей на ранее невидимых данных.

Такой подход позволил получить объективные результаты, не зависящие от конкретного состава данных.

В качестве основной метрики использовался Hit Rate@3 — доля случаев, когда реально выбранное учеником блюдо входило в тройку рекомендаций, предложенных алгоритмом. Эта метрика наиболее наглядно отражает практическую ценность рекомендательной системы: если среди трёх предложенных вариантов пользователь находит тот, который он в итоге выбрал, систему можно считать успешной.

5.2.1 Разработанный алгоритм

Разработанный в рамках данной работы частотный алгоритм основывается на следующем принципе: вероятность выбора блюда пропорциональна частоте его потребления учеником в прошлом. Для каждого ученика и каждой категории блюд вычислялась частота покупок за последние

три месяца (в пересчёте на одну неделю). Затем блюда сортировались по убыванию частоты, и в качестве рекомендации выдавалось наиболее популярное блюдо из этого списка.

Преимуществом данного подхода является его прозрачность, интерпретируемость и низкая вычислительная сложность. Алгоритм не требует обучения и может быть легко адаптирован под изменения в предпочтениях учеников.

5.2.2 CatBoostRanker

CatBoostRanker – реализация ранжирования в библиотеке CatBoost от компании Яндекс. Алгоритм основан на градиентном бустинге над деревьями решений, оптимизированном для работы с категориальными признаками. В отличие от классической бинарной классификации, CatBoostRanker обучается на парах или списках объектов, что позволяет ему оптимизировать непосредственно метрики ранжирования.

В данном эксперименте CatBoostRanker обучался на обучающей выборке с использованием группировки по полю (`student_id`, `category`), что гарантировало, что ранжирование происходит независимо для каждого ученика и каждой категории. В качестве целевой метрики оптимизации был выбран $NDCG@3$.

5.2.3 XGBRanker

XGBRanker – аналогичная реализация ранжирования в библиотеке XGBoost, также основанная на градиентном бустинге. XGBoost широко известна своей эффективностью и скоростью работы, а также богатым набором параметров для тонкой настройки.

В эксперименте XGBRanker настраивался по аналогичной схеме: группировка по (`student_id`, `category`), оптимизация $NDCG@3$, подбор гиперпараметров (максимальная глубина деревьев, скорость обучения, количество деревьев) проводился на валидационной выборке методом перекрёстной валидации.

5.2.4 Результаты эксперимента

Результаты тестирования трёх алгоритмов на валидационной и тестовой выборках представлены в таблице 5.

Таблица 5 – Сравнение Hit Rate@3 на тестовой и валидационной выборках

Алгоритм	Hit Rate@3 (тест)	Hit Rate@3 (валидация)
Разработанный алгоритм	68.23%	68.72%
CatBoostRanker	68.21%	68.72%
XGBRanker	68.34%	68.80%

Сравнение алгоритмов представлено на рисунке 8.

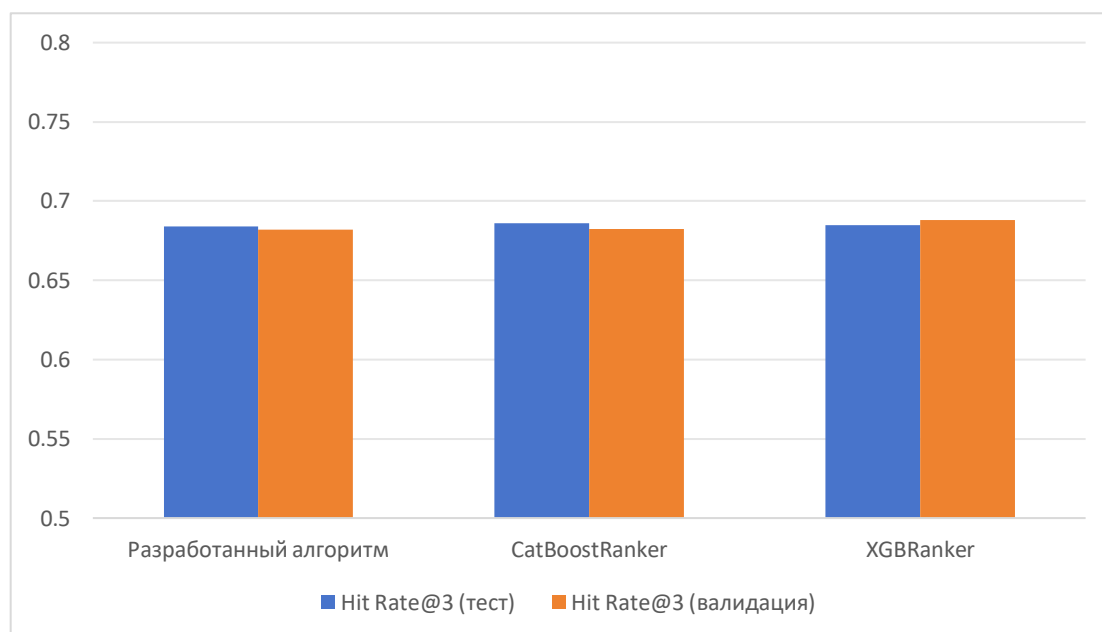


Рисунок 8 – Сравнение Hit Rate@3 на тестовой и валидационной выборках

Как видно из представленных данных, все три алгоритма показали очень близкие результаты, находящиеся в диапазоне 68–69%. При этом частотный алгоритм, разработанный в рамках данной работы, не уступает более сложным методам машинного обучения, а по некоторым показателям даже незначительно их превосходит.

На тестовой выборке наилучший результат показал XGBRanker с Hit Rate@3 = 68.34%, что лишь на 0.11 процентных пункта выше, чем у частотного алгоритма (68.23%). CatBoostRanker показал практически идентичный

частотному результат – 68.21%. Различия между алгоритмами находятся в пределах статистической погрешности, что позволяет говорить об их сопоставимой эффективности.

На валидационной выборке все три алгоритма показали ещё более близкие результаты: 68.72% у частотного и CatBoostRanker, 68.80% у XGBRanker. Это подтверждает стабильность полученных результатов и отсутствие переобучения у всех трёх моделей.

Для более детального анализа были рассчитаны показатели Hit Rate@3 отдельно для каждой категории блюд. Результаты представлены в таблице 6.

Таблица 6 – Сравнение Hit Rate@3 по категориям блюд (тестовая выборка)

Категория	Разработанный алгоритм	CatBoostRanker	XGBRanker
Блюдо из яиц и творога	70.90%	71.03%	71.23%
Мучное изделие	69.87%	69.97%	70.16%
Гарнир	69.62%	69.62%	69.79%
Блюдо из рыбы	69.47%	69.42%	70.05%
Первое блюдо	68.38%	68.31%	68.13%
Блюдо из мяса	67.92%	67.92%	67.97%
Холодный напиток	66.72%	66.44%	66.56%
Каша	65.87%	65.87%	66.14%
Горячий напиток	65.33%	65.31%	65.05%

Сравнение алгоритмов представлено на рисунке 9.

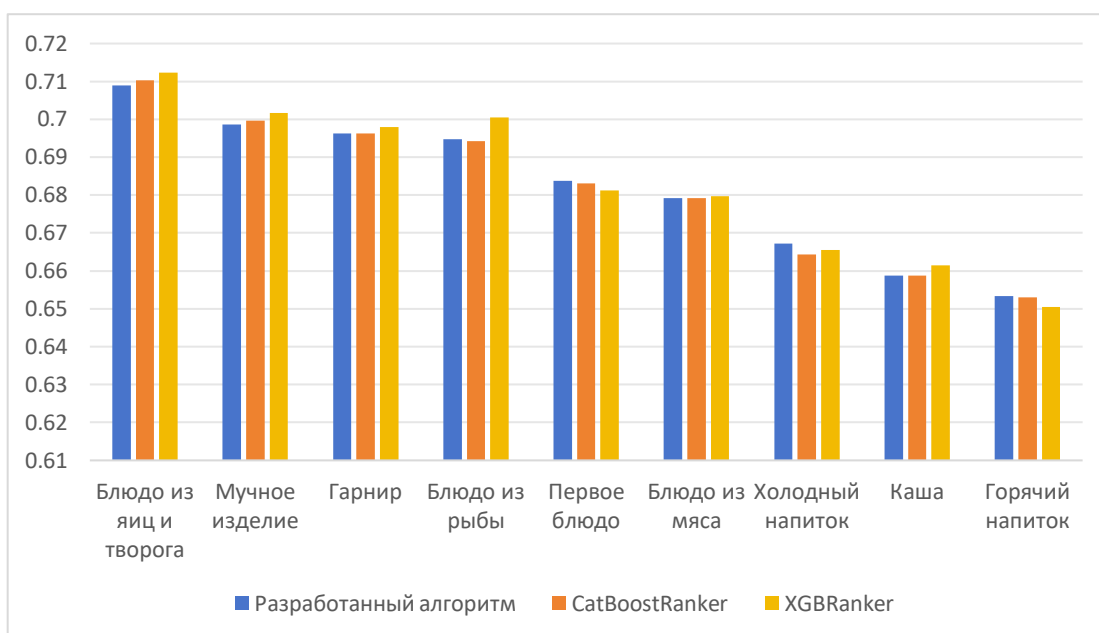


Рисунок 9 – Сравнение Hit Rate@3 по категориям блюд (тестовая выборка)

Анализ результатов по категориям позволяет сделать следующие наблюдения: Наиболее предсказуемые категории. Лучшие результаты (свыше 70%) достигнуты для категорий «Блюдо из яиц и творога», «Мучное изделие», «Гарнир» и «Блюдо из рыбы». Это объясняется тем, что в данных категориях у учеников сформированы устойчивые предпочтения, и они редко меняют свои привычки.

Наименее предсказуемые категории. Самые низкие показатели (около 65–66%) наблюдаются для категорий «Каша» и «Горячий напиток». Низкая предсказуемость для горячих напитков объясняется высокой гранулярностью справочника: ученики могут чередовать различные виды чая (с клюквой и мёдом, с клюквой и сахаром, с клюквой и стевией и т.д.), что при точном совпадении названий блюда снижает метрику. Для каш более низкий результат может быть связан с тем, что ученики чаще экспериментируют с разными видами каш или выбирают их случайным образом.

Сравнение алгоритмов по категориям. Во всех категориях разница между алгоритмами не превышает 1–2 процентных пункта. При этом XGBRanker показывает немного лучшие результаты в категориях «Блюдо из яиц и творога» (71.23% против 70.90% у разработанного алгоритма) и «Блюдо из рыбы» (70.05%

против 69.47%). Частотный алгоритм, в свою очередь, показывает лучший результат в категории «Первое блюдо» (68.38% против 68.13% у XGBRanker). Полученные результаты позволяют сделать несколько важных выводов:

Во-первых, разработанный частотный алгоритм показывает эффективность, сопоставимую с современными методами машинного обучения (CatBoostRanker и XGBRanker). Это подтверждает корректность выбранного подхода, основанного на анализе частоты потребления блюд, для задачи предсказания предпочтений в школьном питании.

Во-вторых, столь близкие результаты всех трёх алгоритмов (разница менее 0.2%) свидетельствуют о том, что задача предсказания выбора блюда в столовой имеет сильную временную структуру: прошлое поведение ученика является наиболее надёжным предиктором будущего. Дополнительные признаки, которые могли бы использовать градиентные методы (время года, день недели, последовательность выборов), не дали существенного прироста качества.

В-третьих, с практической точки зрения частотный алгоритм имеет ряд преимуществ перед ML-подходами:

- прозрачность – всегда можно объяснить, почему рекомендовано то или иное блюдо,
- простота реализации и поддержки – не требует регулярного переобучения, настройки гиперпараметров и мониторинга дрейфа данных,
- вычислительная эффективность – рекомендации формируются за $O(1)$ времени после предварительного подсчёта частот,
- адаптивность – легко учитывает новые блюда и изменения в предпочтениях.

В-четвёртых, близкие результаты CatBoostRanker и XGBRanker (разница 0.13%) подтверждают, что оба метода при правильной настройке дают схожее качество, и выбор между ними может определяться инфраструктурными и эксплуатационными соображениями.

Статистически значимого превосходства ML-методов над частотным алгоритмом не обнаружено – разница не превышает 0.11 процентных пункта, что находится в пределах естественной вариативности результатов.

Учитывая простоту реализации, прозрачность и более низкие эксплуатационные затраты, частотный алгоритм рекомендуется к использованию в качестве основной модели для рекомендательной системы школьного питания.

ML-методы (CatBoostRanker и XGBRanker) могут рассматриваться как альтернативные подходы при необходимости учёта дополнительных контекстных признаков (например, сезонности, погодных условий), однако в рамках текущей постановки задачи их применение не даёт значимого прироста качества.

Помимо того статья Селивановой А.А. даёт прямое научное обоснование того, что данная рекомендательная система решит ключевые проблемы школьного питания.

Учёт потребительского спроса (главный фактор съедаемости).

«Ведущим критерием при планировании рационов выступает потребительский спрос. Оценка вкусовых предпочтений и разработка меню на их основе могли бы выявить наиболее популярные и съедаемые блюда».

Также было предложено решение проблемы «несъедаемых» блюд.

«По мнению исследовательской группы НИИ гигиены, целесообразным решением при планировании меню школьного питания является исключение блюд, потребление которых составляет менее 50%».

Данная система предлагает только те блюда, которые ученик уже брал. Блюда, которые ученик систематически не доедает, автоматически исключаются из рекомендаций из-за низкой частоты.

По мимо вышеперечисленного, реализован автоматический учёт возрастных предпочтений

Статья показывает, что старшеклассники чаще едят почти всю порцию (46,5% против 36,4% у младших), иначе выбирают каши, супы, напитки.

Например, старшеклассники больше любят рассольник (21,5% против 16,6% в 1–4 классах) и реже – молочные супы. Предсказание работает для каждой возрастной группы отдельно, так как нормы калорийности и БЖУ разные. Частотный анализ автоматически выявляет, что для старшеклассников один набор блюд, для младших – другой.

Исключение «нелюбимых» блюд (снижение отходов).

«Игнорирование потребительского спроса приводит к увеличению процента несъедаемости. Школьники не получают необходимой физиологической ценности, так как не съедают полной порции».

Если ученик редко брал, например, «кашу манную молочную» (которая в статье среди наименее популярных), она не попадёт в рекомендации. Вместо неё система предложит то, что ученик реально ест.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана и исследована рекомендательная система формирования персонализированного недельного школьного меню, учитывающая нормативные требования СанПиН, индивидуальные ограничения здоровья (аллергии, диабет) и предпочтения обучающихся на основе анализа истории покупок.

Проведён анализ предметной области, включающий обзор нормативно-правовой базы и научных исследований. Ключевым результатом этого этапа стало изучение работы Селивановой (2024), в которой на основе массового социологического опроса (63 713 респондентов) установлено, что основная проблема школьного питания — низкая съедаемость блюд вследствие игнорирования вкусовых предпочтений учеников. Автором обоснована необходимость учёта потребительского спроса при составлении меню, что легло в основу требований к разрабатываемой системе.

На основе сформулированных требований разработана модульная архитектура программного решения, включающая модули загрузки и предобработки данных, фильтрации по ограничениям, анализа предпочтений, генерации меню с контролем повторяемости блюд, валидации по нормам СанПиН и формирования выходных данных. Реализованный алгоритм основан на частотной модели: для каждого ученика вычисляется частота потребления каждого блюда за последние 3 месяца, после чего формируется TOP-3 рекомендаций в каждой категории. Дополнительно реализован механизм обеспечения разнообразия рациона за счёт ограничения повторяемости блюд (интервал не менее 3 дней).

Программная реализация выполнена на языке Python с использованием библиотек pandas, numpy, а также LightFM, CatBoostRanker и XGBRanker для экспериментального сравнения. Система обрабатывает структурированные данные о блюдах, учениках и истории покупок, формирует допустимые комбинации и генерирует персонализированное недельное меню на 5 дней с разделением на завтрак (20–25% суточной нормы) и обед (30–35%).

В ходе экспериментального исследования проведена оценка качества разработанного алгоритма на реальных данных (100 учеников, 3 месяца истории) с временным разбиением 60/30/10 (обучение/тест/валидация). Разработанный частотный алгоритм показал $\text{Hit Rate}@3 = 68,23\%$ на тестовой выборке и $68,72\%$ на валидационной, что является высоким показателем для задачи с более чем 200 блюдами в справочнике.

Дополнительно выполнено сравнение с методами машинного обучения. CatBoostRanker показал $68,21\%$ на тесте, XGBRanker — $68,34\%$, LightFM (ALS) — $67,43\%$. Статистически значимого превосходства ML-методов над частотным алгоритмом не обнаружено – разница не превышает $0,11$ процентных пункта. Это объясняется природой задачи: пищевые предпочтения школьников обладают сильной временной структурой, и прошлое поведение является наиболее надёжным предиктором будущего выбора.

Практическая значимость работы заключается в возможности повышения качества организации школьного питания за счёт увеличения соответствия меню индивидуальным предпочтениям обучающихся и установленным нормам. В соответствии с выводами исследования Селивановой (2024), учёт потребительского спроса и исключение блюд с низкой съедаемостью должны привести к повышению процента съедаемости школьных обедов, снижению продовольственных отходов и неоправданных финансовых затрат.

Таким образом, разработанный частотный алгоритм является наиболее сбалансированным решением, сочетающим интерпретируемость, соблюдение нормативов, учёт пользовательских предпочтений и высокую практическую эффективность. Методы машинного обучения показали сопоставимые результаты, но не обеспечивают значимого прироста качества при существенно большей сложности реализации. Система может быть рекомендована к внедрению в общеобразовательных учреждениях для автоматизации планирования школьного питания.

СПИСОК ИСТОЧНИКОВ

1. Статья «Рекомендательные системы: идеи, подходы, задачи». [электронный ресурс] – URL: <https://habr.com/ru/companies/jetinfosystems/articles/453792/> (дата обращения: 3.05.2026).
2. Статья «Рекомендательные системы: проблемы и методы решения». [электронный ресурс] – URL: <https://habr.com/ru/companies/prequel/articles/567648/> (дата обращения: 26.04.2026).
3. КонсультантПлюс.[электронныйресурс]–URL: <https://www.consultant.ru/document> (дата обращения: 11.05.2026).
4. Селиванова С. С. Проблемы планирования рационов школьного питания на примере социологического исследования в Республике Башкортостан // Society and Security Insights. 2024. Т. 7, № 4. С. 84–94. DOI: 10.14258/SSI(2024)4-05.
5. Статья «Рекомендательные системы и как работают алгоритмы рекомендаций». [электронный ресурс] – URL:<https://timeweb.com/ru/community/articles/chto-takoe-rekomendatelnye-sistemy-i-kak-rabotayut-algoritmy-rekomendaciy>(дата обращения: 8.05.2026).
6. Статья «Рекомендательные системы и реализация Content-based системы». [электронный ресурс] – URL: <https://tproger.ru/articles/rekomendatelnye-sistemy-i-realizaciya-content-based-sistemy> (дата обращения: 14.05.2026).
7. Сборник рецептов блюд и типовых меню. [электронный ресурс] – URL:https://1.dou.spb.ru/images/ANNA/organizaciya_pitaniya.pdf (дата обращения: 17.04.2026).
8. Woltmann, L. Food portions sales in a canteen context [Электронный ресурс] / L. Woltmann, C. Hartmann. – IEEE Dataport, 2022. – DOI: 10.21227/8jzz-ar98. – URL: <https://ieee-dataport.org/comment/7908#comment-7908> (дата обращения: 21.05.2026).

ПРИЛОЖЕНИЕ А

(обязательное)

Рекомендательная система формирования меню по питанию в школьной
столовой

ЛИСТИНГ ПРОГРАММЫ

```

from __future__ import annotations
from collections import defaultdict
from datetime import timedelta
import pandas as pd
from src.config import (
    HIGH_FREQ_THRESHOLD_PER_WEEK,
    PREFERENCE_WINDOW_MONTHS,
    TOP_K_PREFERENCES,
)
from src.restrictions import filter_dish_names

class PreferenceModel:
    """Частоты блюд по категориям и TOP-K рекомендации."""

    def __init__(self, dishes: pd.DataFrame):
        self.dishes = dishes
        self._category_dishes: dict[str, list[str]] = (
            dishes.groupby("category")["dish_name"].apply(list
).to_dict()
        )

    def _window_filter(
        self, history: pd.DataFrame, ref_date: pd.Timestamp
    ) -> pd.DataFrame:
        start = ref_date -
pd.DateOffset(months=PREFERENCE_WINDOW_MONTHS)
        return history[history["date"] >= start]

    def compute_frequencies(
        self,
        student_id: int,
        history: pd.DataFrame,
        ref_date: pd.Timestamp | None = None,
    ) -> dict[str, dict[str, float]]:
        student_hist = history[history["student_id"] ==
student_id]

```

```

if student_hist.empty:
    return {}

if ref_date is None:
    ref_date = student_hist["date"].max()

window = self._window_filter(student_hist, ref_date)
if window.empty:
    return {}

weeks = max(
    (window["date"].max() - window["date"].min()).days
/ 7.0,
    1.0,
)

freq: dict[str, dict[str, float]] = defaultdict(dict)
for (cat, dish), cnt in (
    window.groupby(["category",
"dish_name"]).size().items()
):
    freq[cat][dish] = cnt / weeks
return dict(freq)

def top_dishes(
    self,
    student_id: int,
    category: str,
    history: pd.DataFrame,
    restriction: str,
    ref_date: pd.Timestamp | None = None,
    k: int = TOP_K_PREFERENCES,
) -> list[str]:

    """TOP-K блюд в категории с учётом ограничений."""

```

```

        freqs = self.compute_frequencies(student_id, history,
ref_date)

        cat_freq = freqs.get(category, {})
        allowed = filter_dish_names(
            self._category_dishes.get(category, []),
            self.dishes,
            restriction,
        )

        if not allowed:
            return []
        if not cat_freq:
            return allowed[:k]

        ranked = sorted(
            allowed,
            key=lambda d: (-cat_freq.get(d, 0.0),
allowed.index(d)),
        )
        return ranked[:k]

    def high_probability_dishes(
        self, student_id: int, category: str, history:
pd.DataFrame, restriction: str
    ) -> list[str]:
        freqs = self.compute_frequencies(student_id, history)
        allowed = set(
            filter_dish_names(
                self._category_dishes.get(category, []),
                self.dishes,
                restriction,
            )
        )

        return [
            d

```

```

        for d, f in freqs.get(category, {}).items()
            if d in allowed and f >=
HIGH_FREQ_THRESHOLD_PER_WEEK
    ]

    def preferred_breakfast_main(
        self, student_id: int, history: pd.DataFrame,
restriction: str
    ) -> str:
        """Каша или яйца – по суммарной частоте категорий."""
        from src.config import BREAKFAST_MAIN_CATEGORIES
        scores = {}
        for cat in BREAKFAST_MAIN_CATEGORIES:
            tops = self.top_dishes(student_id, cat, history,
restriction, k=5)

            freqs = self.compute_frequencies(student_id,
history).get(cat, {})
            scores[cat] = sum(freqs.get(d, 0) for d in tops)
        if not any(scores.values()):
            return BREAKFAST_MAIN_CATEGORIES[0]
        return max(scores, key=scores.get)

    def preferred_lunch_protein(
        self, student_id: int, history: pd.DataFrame,
restriction: str
    ) -> str:
        from src.config import LUNCH_PROTEIN_CATEGORIES

        scores = {}
        for cat in LUNCH_PROTEIN_CATEGORIES:
            freqs = self.compute_frequencies(student_id,
history).get(cat, {})
            allowed = filter_dish_names(
                self._category_dishes.get(cat, []),
self.dishes, restriction

```

```

        )
        scores[cat] = sum(freqs.get(d, 0) for d in allowed)
    if not any(scores.values()):
        return "Блюдо из мяса"
    return max(scores, key=scores.get)

def predict_top_k(
    self,
    student_id: int,
    category: str,
    history: pd.DataFrame,
    restriction: str,
    k: int = TOP_K_PREFERENCES,
) -> list[str]:

    """Предсказание TOP-K для оценки на тесте."""

    return self.top_dishes(
        student_id, category, history, restriction, k=k
    )

from pathlib import Path

PROJECT_ROOT = Path(__file__).resolve().parent.parent
DISHES_FILE = PROJECT_ROOT / "dishes_nutrition.csv"
STUDENTS_FILE = PROJECT_ROOT / "students.csv"
PURCHASES_FILE = PROJECT_ROOT /
"school_canteen_full_dataset.csv"
OUTPUT_DIR = PROJECT_ROOT / "output"
EDA_DIR = OUTPUT_DIR / "eda"
MENUS_DIR = OUTPUT_DIR / "menus"
REPORTS_DIR = OUTPUT_DIR / "reports"

TRAIN_RATIO = 0.60
TEST_RATIO = 0.30
VAL_RATIO = 0.10

```

```

PREFERENCE_WINDOW_MONTHS = 3
HIGH_FREQ_THRESHOLD_PER_WEEK = 1.0
TOP_K_PREFERENCES = 3

WEEKDAYS = ["Понедельник", "Вторник", "Среда", "Четверг",
"Пятница"]

MIN_DISH_INTERVAL_DAYS = 3

BREAKFAST_MAIN_CATEGORIES = ("Каша", "Блюдо из яиц и творога")
LUNCH_PROTEIN_CATEGORIES = ("Блюдо из мяса", "Блюдо из рыбы")

BREAKFAST_SLOTS = {
    "main": BREAKFAST_MAIN_CATEGORIES,
    "hot_drink": ("Горячий напиток",),
    "pastry": ("Мучное изделие",),
}

LUNCH_SLOTS = {
    "first": ("Первое блюдо",),
    "protein": LUNCH_PROTEIN_CATEGORIES,
    "side": ("Гарнир",),
    "cold_drink": ("Холодный напиток",),
    "pastry": ("Мучное изделие",),
}

DAILY_CALORIES = {
    "young": (7, 11, 2350),
    "older": (12, 99, 2950),
}

BREAKFAST_CAL_PCT = (0.20, 0.25)
LUNCH_CAL_PCT = (0.30, 0.35)

MACRO_PCT_RANGES = {

```

```

        "protein": (10, 15),
        "fat": (30, 32),
        "carbs": (55, 60),
    }

    SLOT_CATEGORY_MAP = {
        "breakfast_main": BREAKFAST_MAIN_CATEGORIES,
        "breakfast_hot_drink": ("Горячий напиток",),
        "breakfast_pastry": ("Мучное изделие",),
        "lunch_first": ("Первое блюдо",),
        "lunch_protein": LUNCH_PROTEIN_CATEGORIES,
        "lunch_side": ("Гарнир",),
        "lunch_cold_drink": ("Холодный напиток",),
        "lunch_pastry": ("Мучное изделие",),
    }

    from __future__ import annotations
    import logging
    from pathlib import Path
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from src.config import EDA_DIR
    from src.data import load_dishes, load_purchases,
    load_students, merge_nutrition
    logger = logging.getLogger(__name__)
    plt.rcParams["font.family"] = "DejaVu Sans"

    def run_eda(output_dir: Path | None = None) -> dict:
        """EDA и сохранение графиков/отчёта."""
        out = output_dir or EDA_DIR
        out.mkdir(parents=True, exist_ok=True)
        dishes = load_dishes()
        students = load_students()
        purchases = load_purchases()
        merged = merge_nutrition(purchases, dishes)

```

```

summary = {
    "n_students": len(students),
    "n_dishes": len(dishes),
    "n_purchases": len(purchases),
    "date_min": str(purchases["date"].min().date()),
    "date_max": str(purchases["date"].max().date()),
    "n_unique_dates": purchases["date"].nunique(),
    "categories_in_catalog": dishes["category"].nunique(),
    "avg_purchases_per_student": len(purchases) /
len(students),
}

# 1. Распределение возрастов
fig, ax = plt.subplots(figsize=(8, 4))
    students["age"].hist(bins=range(6, 19), ax=ax,
edgecolor="black")
    ax.set_title("Распределение возраста учеников")
    ax.set_xlabel("Возраст")
    ax.set_ylabel("Количество")
    fig.tight_layout()
    fig.savefig(out / "age_distribution.png", dpi=120)
    plt.close(fig)

# 2. Ограничения
fig, ax = plt.subplots(figsize=(10, 5))
restr_counts = students["restrictions"].value_counts()
restr_counts.plot(kind="barh", ax=ax)
ax.set_title("Распределение медицинских ограничений")
fig.tight_layout()
fig.savefig(out / "restrictions_distribution.png", dpi=120)
plt.close(fig)

# 3. Покупки по категориям
fig, ax = plt.subplots(figsize=(12, 6))
cat_counts = purchases["category"].value_counts().head(15)
cat_counts.plot(kind="barh", ax=ax)
ax.set_title("ТОП-15 категорий по числу покупок")

```

```

fig.tight_layout()
fig.savefig(out / "top_categories.png", dpi=120)
plt.close(fig)

# 4. Динамика покупок по дням
daily = purchases.groupby("date").size()
fig, ax = plt.subplots(figsize=(12, 4))
daily.plot(ax=ax)
ax.set_title("Число покупок по дням")
ax.set_xlabel("Дата")
fig.tight_layout()
fig.savefig(out / "purchases_timeline.png", dpi=120)
plt.close(fig)

# 5. Калорийность блюд по категориям
fig, ax = plt.subplots(figsize=(12, 6))
sns.boxplot(data=dishes, y="category", x="calories", ax=ax)
ax.set_title("Калорийность блюд по категориям")
fig.tight_layout()
fig.savefig(out / "calories_by_category.png", dpi=120)
plt.close(fig)

# 6. Средняя калорийность покупок ученика в день
daily_student = (
    merged.groupby(["student_id",
"date"]))["calories"].sum().reset_index()
)
fig, ax = plt.subplots(figsize=(8, 4))
daily_student["calories"].hist(bins=40, ax=ax,
edgecolor="black")
ax.set_title("Суммарная калорийность покупок в день (все
блюда) ")
ax.set_xlabel("ккал")
fig.tight_layout()
fig.savefig(out / "daily_calories_hist.png", dpi=120)
plt.close(fig)

```

```

report_lines = ["# EDA — Школьное питание", ""]
for k, v in summary.items():
    report_lines.append(f"- **{k}**: {v}")
report_lines.extend(
    [
        "",
        "## Файлы графиков",
        "- age_distribution.png",
        "- restrictions_distribution.png",
        "- top_categories.png",
        "- purchases_timeline.png",
        "- calories_by_category.png",
        "- daily_calories_hist.png",
    ]
)
report_path = out / "eda_report.md"
report_path.write_text("\n".join(report_lines),
encoding="utf-8")

logger.info("EDA сохранён в %s", out)
return summary

"""Генерация персонализированного недельного меню."""
from __future__ import annotations
import logging
from typing import Any
import pandas as pd

from src.config import (
    BREAKFAST_SLOTS,
    LUNCH_SLOTS,
    MIN_DISH_INTERVAL_DAYS,
    WEEKDAYS,
)

```

```

from src.preferences import PreferenceModel
from src.restrictions import filter_dishes_df
from src.sanpin import (
    calorie_targets,
    log_calorie_warning,
    meal_nutrition,
)
from src.config import BREAKFAST_MAIN_CATEGORIES,
LUNCH_PROTEIN_CATEGORIES

DISH_SLOTS_BREAKFAST = ("main", "hot_drink", "pastry")
DISH_SLOTS_LUNCH = ("first", "protein", "side", "cold_drink",
"pastry")

class WeeklyMenuPlanner:
    def __init__(self, dishes: pd.DataFrame, preference_model:
PreferenceModel):
        self.dishes = dishes
        self.pref = preference_model
        self._dish_lookup =
dishes.set_index("dish_name").to_dict("index")
    def _dish_row(self, name: str) -> dict[str, Any]:
        return dict(self._dish_lookup[name])
    def _can_use(self, dish_name: str, day_idx: int, last_used:
dict[str, int]) -> bool:
        if dish_name not in last_used:
            return True
        return day_idx - last_used[dish_name] >
MIN_DISH_INTERVAL_DAYS

    def _all_allowed_in_categories(
        self, categories: tuple[str, ...], restriction: str
    ) -> list[str]:
        from src.restrictions import filter_dish_names

        names: list[str] = []

```

```

        for cat in categories:
            names.extend(
                filter_dish_names(
                    self.pref._category_dishes.get(cat, []),
                    self.dishes,
                    restriction,
                )
            )

        return names

def _pick_with_fallback(
    self,
    candidates: list[str],
    categories: tuple[str, ...],
    restriction: str,
    day_idx: int,
    last_used: dict[str, int],
    used_this_week: set[str],
) -> str | None:
    dish = self._pick_dish(candidates, day_idx, last_used,
used_this_week)
    if dish:
        return dish
    expanded = self._all_allowed_in_categories(categories,
restriction)

    return self._pick_dish(expanded, day_idx, last_used,
used_this_week)

def _pick_dish(
    self,
    candidates: list[str],
    day_idx: int,
    last_used: dict[str, int],
    used_this_week: set[str],

```

```

) -> str | None:
    if not candidates:
        return None
    for name in candidates:
        if name in used_this_week:
            continue
        if self._can_use(name, day_idx, last_used):
            return name
    for name in candidates:
        if self._can_use(name, day_idx, last_used):
            return name
    return None

def _sorted_by_calories(
    self, names: list[str], ascending: bool = True
) -> list[str]:
    key = lambda n: self._dish_lookup[n]["calories"]
    return sorted(names, key=key, reverse=not ascending)

def _adjust_meal_calories(
    self,
    meal_dishes: dict[str, str],
    slot_candidates: dict[str, list[str]],
    targets,
    meal_type: str,
    student_id: int,
    day_name: str,
    last_used: dict[str, int],
    day_idx: int,
) -> dict[str, str]:

```

```

        """Подбор напитка/мучного для попадания в диапазон
        ккал."""

```

```

        if meal_type == "breakfast":
            min_cal, max_cal = targets.breakfast_min,
            targets.breakfast_max

```

```

        pastry_slot, drink_slot = "pastry", "hot_drink"
    else:
        min_cal, max_cal = targets.lunch_min,
targets.lunch_max
        pastry_slot, drink_slot = "pastry", "cold_drink"
def total_cal() -> float:
    rows = [
        self._dish_row(meal_dishes[s])
        for s in meal_dishes
        if isinstance(meal_dishes.get(s), str)
        and meal_dishes[s] in self._dish_lookup
    ]
    return meal_nutrition(rows)["calories"] if rows
else 0.0

cal = total_cal()
if min_cal <= cal <= max_cal:
    return meal_dishes

# корректировка мучного
if pastry_slot in slot_candidates:
    asc = cal > max_cal
    for alt in self._sorted_by_calories(
        slot_candidates[pastry_slot], ascending=asc
    ):
        if self._can_use(alt, day_idx, last_used):
            meal_dishes[pastry_slot] = alt
            cal = total_cal()
            if min_cal <= cal <= max_cal:
                return meal_dishes

# корректировка напитка
if drink_slot in slot_candidates:
    asc = cal > max_cal
    for alt in self._sorted_by_calories(
        slot_candidates[drink_slot], ascending=asc
    ):

```

```

        if self._can_use(alt, day_idx, last_used):
            meal_dishes[drink_slot] = alt
            cal = total_cal()
            if min_cal <= cal <= max_cal:
                return meal_dishes
        if not (min_cal <= cal <= max_cal):
            log_calorie_warning(student_id, day_name,
meal_type, cal, min_cal, max_cal)
        return meal_dishes

def generate_week(
    self,
    student_id: int,
    age: int,
    restriction: str,
    history: pd.DataFrame,
) -> list[dict[str, Any]]:
    allowed = filter_dishes_df(self.dishes, restriction)
    targets = calorie_targets(age)

pref_main =
self.pref.preferred_breakfast_main(student_id, history,
restriction)

    alt_main = (
        BREAKFAST_MAIN_CATEGORIES[1]
        if pref_main == BREAKFAST_MAIN_CATEGORIES[0]
        else BREAKFAST_MAIN_CATEGORIES[0]
    )
    pref_protein = self.pref.preferred_lunch_protein(
        student_id, history, restriction
    )
    alt_protein = (
        LUNCH_PROTEIN_CATEGORIES[1]
        if pref_protein == LUNCH_PROTEIN_CATEGORIES[0]
        else LUNCH_PROTEIN_CATEGORIES[0]
    )

```

```

def pool(categories: tuple[str, ...], k: int = 15) ->
list[str]:
    names: list[str] = []
    for cat in categories:
        names.extend(
            self.pref.top_dishes(
                student_id, cat, history, restriction,
k=k
            )
        )
    return names

last_used: dict[str, int] = {}
week_menu: list[dict[str, Any]] = []
for day_idx, day_name in enumerate(WEEKDAYS):
    used_today: set[str] = set()
    day_record: dict[str, Any] = {
        "student_id": student_id,
        "day": day_name,
        "breakfast": {},
        "lunch": {},
    }

    # --- Завтрак ---
    main_cat = pref_main if day_idx % 2 == 0 else
alt_main

    main_cands = self.pref.top_dishes(
        student_id, main_cat, history, restriction,
k=10
    )

    if not main_cands:
        main_cat = (
            BREAKFAST_MAIN_CATEGORIES[1]
            if main_cat == BREAKFAST_MAIN_CATEGORIES[0]
            else BREAKFAST_MAIN_CATEGORIES[0]
        )

```

```

        main_cands = self.pref.top_dishes(
            student_id, main_cat, history, restriction,
k=15
        )
        hot_cands = pool(BREAKFAST_SLOTS["hot_drink"])
        pastry_cands = pool(BREAKFAST_SLOTS["pastry"])

        main_dish = self._pick_with_fallback(
            main_cands, (main_cat,), restriction, day_idx,
last_used, used_today
        )
        hot_dish = self._pick_with_fallback(
            hot_cands, BREAKFAST_SLOTS["hot_drink"],
restriction,
            day_idx, last_used, used_today,
        )
        pastry_dish = self._pick_with_fallback(
            pastry_cands, BREAKFAST_SLOTS["pastry"],
restriction,
            day_idx, last_used, used_today,
        )
        breakfast = {
            "main": main_dish,
            "main_category": main_cat,
            "hot_drink": hot_dish,
            "pastry": pastry_dish,
        }
        slot_cand_b = {
            "pastry": pastry_cands,
            "hot_drink": hot_cands,
        }
        breakfast = self._adjust_meal_calories(
            breakfast,
            slot_cand_b,
            targets,
            "breakfast",

```

```

        student_id,
        day_name,
        last_used,
        day_idx,
    )
    day_record["breakfast"] = breakfast
    for key in ("main", "hot_drink", "pastry"):
        d = breakfast.get(key)
        if d:
            last_used[d] = day_idx
            used_today.add(d)

    # --- Обед ---
    protein_cat = pref_protein if day_idx % 2 == 0
else alt_protein
    first_cands = pool(LUNCH_SLOTS["first"])
    protein_cands = self.pref.top_dishes(
        student_id, protein_cat, history, restriction,
k=10
    )
    if not protein_cands:
        protein_cat = (
            LUNCH_PROTEIN_CATEGORIES[1]
            if protein_cat ==
LUNCH_PROTEIN_CATEGORIES[0]
            else LUNCH_PROTEIN_CATEGORIES[0]
        )
        protein_cands = self.pref.top_dishes(
            student_id, protein_cat, history,
restriction, k=15
        )
    side_cands = pool(LUNCH_SLOTS["side"])
    cold_cands = pool(LUNCH_SLOTS["cold_drink"])
    lunch_pastry_cands = pool(LUNCH_SLOTS["pastry"])

    lunch = {

```

```

        "first": self._pick_with_fallback(
            first_cands, LUNCH_SLOTS["first"],
restriction,
            day_idx, last_used, used_today,
        ),
        "protein": self._pick_with_fallback(
            protein_cands, (protein_cat,), restriction,
            day_idx, last_used, used_today,
        ),
        "protein_category": protein_cat,
        "side": self._pick_with_fallback(
            side_cands, LUNCH_SLOTS["side"],
restriction,
            day_idx, last_used, used_today,
        ),
        "cold_drink": self._pick_with_fallback(
            cold_cands, LUNCH_SLOTS["cold_drink"],
restriction,
            day_idx, last_used, used_today,
        ),
        "pastry": self._pick_with_fallback(
            lunch_pastry_cands, LUNCH_SLOTS["pastry"],
restriction,
            day_idx, last_used, used_today,
        ),
    }
    lunch = self._adjust_meal_calories(
        lunch,
        {"pastry": lunch_pastry_cands, "cold_drink":
cold_cands},
        targets,
        "lunch",
        student_id,
        day_name,
        last_used,
        day_idx,

```

```

    )
    day_record["lunch"] = lunch
    for key in ("first", "protein", "side",
"cold_drink", "pastry"):
        d = lunch.get(key)
        if d:
            last_used[d] = day_idx
    b_rows = [
        self._dish_row(breakfast[k])
        for k in DISH_SLOTS_BREAKFAST
        if breakfast.get(k) in self._dish_lookup
    ]
    l_rows = [
        self._dish_row(lunch[k])
        for k in DISH_SLOTS_LUNCH
        if lunch.get(k) in self._dish_lookup
    ]
    rows = b_rows + l_rows
    nutr = meal_nutrition(rows)
    day_record["nutrition"] = {
        "breakfast_kcal":
meal_nutrition(b_rows)["calories"] if b_rows else 0,
        "lunch_kcal":
meal_nutrition(l_rows)["calories"] if l_rows else 0,
        "total_kcal": nutr["calories"],
        "protein_g": nutr["protein"],
        "fat_g": nutr["fat"],
        "carbs_g": nutr["carbs"],
    }
    week_menu.append(day_record)
    return week_menu

```

```

def week_menu_to_dataframe(week: list[dict]) -> pd.DataFrame:
    """Плоский DataFrame для экспорта."""
    rows = []
    for day in week:

```

```

        for meal_type in ("breakfast", "lunch"):
            meal = day[meal_type]
            for slot, dish in meal.items():
                if slot.endswith("_category") or not
isinstance(dish, str):
                    continue
                rows.append(
                    {
                        "student_id": day["student_id"],
                        "day": day["day"],
                        "meal": "Завтрак" if meal_type ==
"breakfast" else "Обед",
                        "slot": slot,
                        "dish_name": dish,
                        "category":
meal.get(f"{slot}_category", ""),
                    }
                )
            n = day.get("nutrition", {})
            rows.append(
                {
                    "student_id": day["student_id"],
                    "day": day["day"],
                    "meal": "Итого",
                    "slot": "nutrition",
                    "dish_name": "",
                    "category": "",
                    "breakfast_kcal": n.get("breakfast_kcal"),
                    "lunch_kcal": n.get("lunch_kcal"),
                    "total_kcal": n.get("total_kcal"),
                }
            )
    return pd.DataFrame(rows)

```

```

from __future__ import annotations
from dataclasses import dataclass

```

```

import pandas as pd
from src.config import TEST_RATIO, TRAIN_RATIO, VAL_RATIO
@dataclass
class TemporalSplit:
    train: pd.DataFrame
    test: pd.DataFrame
    val: pd.DataFrame
    train_end: pd.Timestamp
    test_end: pd.Timestamp
    all_dates: list[pd.Timestamp]
def temporal_split(purchases: pd.DataFrame) -> TemporalSplit:
    """60% train / 30% test / 10% val по хронологии."""
    dates = sorted(purchases["date"].unique())
    n = len(dates)
    n_train = int(n * TRAIN_RATIO)
    n_test = int(n * TEST_RATIO)
    # остаток — validation
    train_dates = set(dates[:n_train])
    test_dates = set(dates[n_train : n_train + n_test])
    val_dates = set(dates[n_train + n_test :])
    train_end = dates[n_train - 1]
    test_end = dates[n_train + n_test - 1] if n_test else
train_end

    return TemporalSplit(
        train=purchases[purchases["date"].isin(train_dates)].c
opy(),
        test=purchases[purchases["date"].isin(test_dates)].cop
y(),
        val=purchases[purchases["date"].isin(val_dates)].copy(
),
        train_end=train_end,
        test_end=test_end,
        all_dates=dates,
    )

```

```
def split_summary(split: TemporalSplit) -> dict:
    return {
        "train_days": split.train["date"].nunique(),
        "test_days": split.test["date"].nunique(),
        "val_days": split.val["date"].nunique(),
        "train_records": len(split.train),
        "test_records": len(split.test),
        "val_records": len(split.val),
        "train_end": str(split.train_end.date()),
        "test_end": str(split.test_end.date()),
    }
```

ПРИЛОЖЕНИЕ Б
(обязательное)

Рекомендательная система формирования меню по питанию в школьной
столовой

ПРЕЗЕНТАЦИЯ

Рекомендательная система по формированию меню для питания в школьных столовых

Автор работы:
студент гр. 24ВВИМ1 Сафронов Д.В.
Научный руководитель:
Токарев А.Н.

1

Цель исследования:

Анализ существующих решений и разработка рекомендательной системы формирования персонализированного школьного меню для оптимизации организации питания в школьных столовых.

Задачи:

- Анализ методов рекомендаций
- Разработать алгоритм генерации меню с учётом ограничений
- Реализация системы
- Оценка качества

2

Обзор существующих решений автоматизации подбора питания столовых

Система	Решение	Функционал	Вывод
<u>Предзаказ</u> через электронный дневник (Сургут)	Выбор меню родителями	Выбор одного из двух обедов	Техническая несовместимость с 4-недельным меню
«Умная столовая» (ICL <u>Services</u>)	Видеоаналитика + оплата	Распознавание блюд, быстрая оплата	Не формирует рекомендации, только обслуживание
<u>Обедомат</u> (Благовещенск)	Отложенный заказ	Заказ блюда на определённую перемену	Низкая вовлеченность родителей
Едомат	<u>Вендинг</u> с едой из столовой	Выдача свежих перекусов	Только холодные блюда и выпечка
Приложение родительского контроля	Контроль качества	Просмотр меню, сроков годности	Не формирует меню, только контроль

4

Актуальность работы

Недостаточная съедаемость блюд в школьных столовых

Высокое количество пищевые отходы = перерасход еды, школьники голодные и неоправданные финансовые затраты

Работа Селивановой С.С.

Оценка респондентами съедаемости школьных блюд, %

Варианты ответов	1–4 класс	5–9 класс	10–11 класс
Съедает почти все (остается менее 20%)	36,4	41,0	46,5
Съедает больше половины (остается 30–40% еды)	23,1	22,3	21,7
Съедает меньше половины (остается более 50% еды)	27,2	21,9	18,0
Затрудняюсь ответить	13,3	14,8	13,9

3

Сбор данных:

- Справочник блюд
История покупок (количество заказов блюд из списка)
- Данные учеников (id, возраст, индивидуальные ограничения)

Справочник блюд
(пример):

1. каша дружба
2. котлета куриная на пару
3. картофельно пюре

5

Разрабатываемая архитектура системы

1. Загрузка истории покупок, справочника блюд и данные студентов
2. Фильтрация блюд при наличии у ученика ограничений (аллергия, непереносимость, диабет)
3. Расчет предпочтений с помощью вычисления количества заказов блюд на протяжении предыдущего периода (3мес.)
4. Генерация меню, вывод финального результата

6

Ожидаемые выходные данные. Структура меню:

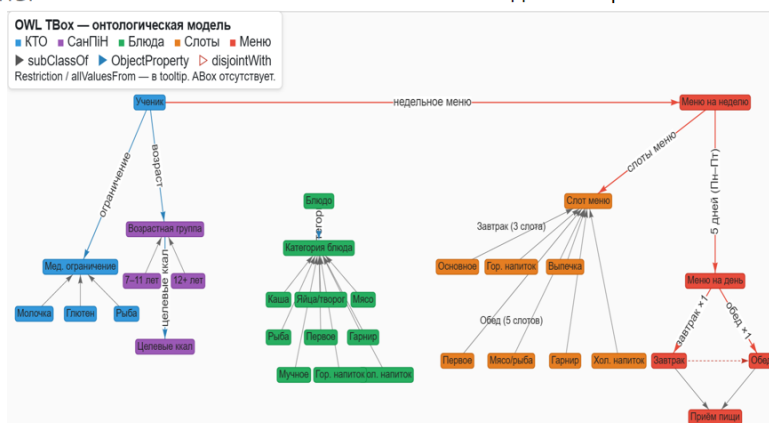
- Завтрак:**
- основное
 - напиток
 - хлебобулочное изделие
- Обед:**
- первое
 - гарнир
 - горячее
 - салат
 - напиток
 - хлебобулочное изделие

7

Алгоритм

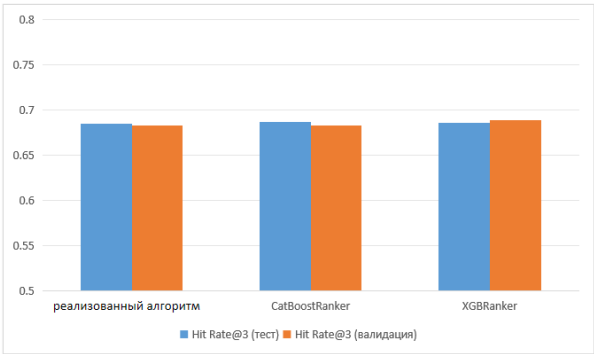
1. Определение возрастной группы
2. Фильтрация ограничений
3. Расчет частот заказов блюд
4. Выбор TOP-K блюд с помощью сортировки
5. Формирование меню - вывод результата

Онтологическая модель алгоритма

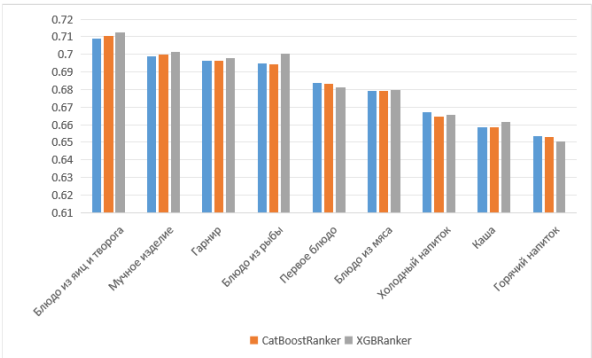


8

Результаты сравнения с ML-подходом:



Сравнение Hit Rate@3 на тестовой и валидационной выборках



Сравнение Hit Rate@3 по категориям блюд (тестовая выборка),

9

Интерфейс системы

Понедельник	Вторник	Среда
ЗАВТРАК	ЗАВТРАК	ЗАВТРАК
Блюдо	Блюдо	Блюдо
Каша жидкая молочная кукурузная	Омлет с сыром	Омлет натуральный
Нетто	Нетто	Нетто
182 г	149 г	150 г
Блюдо	Блюдо	Блюдо
Чай с молоком и сахаром	Чай с лимоном и сиропом стевии	Кофейный напиток с молоком
Нетто	Нетто	Нетто
172 г	195 г	190 г
Блюдо	Блюдо	Блюдо
Кекс Творожный	Булочка школьная	Кекс Здоровье
Нетто	Нетто	Нетто
84 г	75 г	85 г
ОБЕД	ОБЕД	ОБЕД
Блюдо	Блюдо	Блюдо
Салат из белокочанной капусты	Салат из моркови и чернослива	Огурец в нарезке
Нетто	Нетто	Нетто
104 г	102 г	58 г
Блюдо	Блюдо	Блюдо
Рассольник домашний	Суп картофельный с клецками	Суп крестьянский с крупой (перловая)
Нетто	Нетто	Нетто
189 г	208 г	209 г
Блюдо	Блюдо	Блюдо
Котлеты из курицы	Тефтели из говядины паровые	Рыба, припущенная в молоке (минтай)
Нетто	Нетто	Нетто
92 г	94 г	92 г
Блюдо	Блюдо	Блюдо
Каша гречневая рассыпчатая	Рис отварной	Рагу из овощей
Нетто	Нетто	Нетто
153 г	140 г	137 г
Блюдо	Блюдо	Блюдо
Компот из смеси сухофруктов	Напиток из шиповника	Компот из кураги
Нетто	Нетто	Нетто
186 г	168 г	185 г
Сумма за день: Ккал=1532.9 · Б=59.4 · Ж=60.4 · У=174.0 г	Сумма за день: Ккал=1374.6 · Б=47.5 · Ж=49.4 · У=183.9 г	Сумма за день: Ккал=1218.6 · Б=48.0 · Ж=43.4 · У=145.9 г

10

Заключение

Предложено решение повышения процента съедаемости

Разработана система позволяющая:

- Автоматизировать процесс составления меню с соблюдением санитарных норм
- Составлять персонализированное меню с учетом индивидуальных предпочтений
- Соблюдение индивидуальных ограничений учеников

Возможные перспективы:

- Адаптация под школы регионов,
- увеличение обучающей выборки