Демич А.А., Митянина А.В. Разработка интеллектуального помощника для студентов и сотрудников ВУЗА// Международный журнал информационных технологий и энергоэффективности.– 2026. –Т. 11 № 6(68) Ч. 1 с. 91–100

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/



УДК 004.89

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПОМОЩНИКА ДЛЯ СТУДЕНТОВ И СОТРУДНИКОВ ВУЗА

[1] **Демич А.А., Митянина А.В.**
ФГБОУ ВО "ЧЕЛЯБИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ", Челябинск, Россия (454001, Челябинская область, город Челябинск, ул Братьев Кашириных, д. 129), e-mail: [1]demich-2002@mail.ru

В условиях цифровизации высшего образования возрастает объём информации, с которой ежедневно взаимодействуют студенты и сотрудники вузов. В статье рассматривается проблема фрагментированности цифровых сервисов университета, затрудняющая доступ обучающихся к оперативной информации и создающая избыточную нагрузку на административный персонал. Для решения данной проблемы спроектирован и реализован интеллектуальный помощник с микросервисной архитектурой, объединяющий в себе модуль диалогового поиска по нормативной документации (на базе технологии RAG) и модуль предиктивной аналитики успеваемости.

Ключевые слова: Интеллектуальные системы, искусственный интеллект, RAG-архитектура, машинное обучение, предиктивная аналитика, данные об успеваемости, микросервисная архитектура.

DEVELOPMENT OF AN INTELLIGENT ASSISTANT FOR UNIVERSITY STUDENTS AND STAFF

[1] Demich A. A., Mityanina A. V.
CHELYABINSK STATE UNIVERSITY, Chelyabinsk, Russia (454001, Chelyabinsk region, Chelyabinsk city, Bratyev Kashirinykh st., 129,), e-mail: [1]demich-2002@mail.ru

As higher education undergoes digitalisation, the volume of information that students and university staff interact with on a daily basis is increasing. This article examines the problem of fragmentation in university digital services, which hinders students' access to up-to-date information and places an excessive burden on administrative staff. To address this issue, a microservice architecture for an intelligent assistant has been designed and implemented, combining a dialogue-based search module for regulatory documentation (based on RAG technology) and a predictive analytics module for academic performance.

Keywords: Intelligent systems, artificial intelligence, RAG architecture, machine learning, predictive analytics, educational data, microservices architecture.

Современные университеты активно внедряют цифровые технологии для сопровождения образовательного процесса. Тем не менее, на практике информационная инфраструктура вуза часто представляет собой совокупность слабо связанных между собой систем: электронных журналов, личных кабинетов, корпоративных информационных систем и хранилищ нормативных документов. [1] В контексте высшего медицинского университета (на примере Южно-Уральского государственного медицинского университета) эти процессы приобретают особенно высокую значимость. Учебный процесс студента-медика отличается высокой интенсивностью, строгой регламентацией и колоссальным объемом сложного материала по дисциплинам.

В этих условиях выявляются две критические проблемы, препятствующие эффективному управлению образовательным процессом:

1. Информационная перегрузка и фрагментация среды. Нормативная документация (правила ликвидации задолженностей, порядки прохождения практик) хранится в виде неструктурированных файлов на официальных порталах, тогда как оперативные данные (расписание, текущие оценки) находятся в корпоративной информационной системе (КИС) «1С: Университет». Из-за необходимости обращаться к различным источникам студенты тратят значительное время на поиск базовой информации вместо фокусировки на учебе. Не находя нужный ответ, студентам приходиться обращаться к административному персоналу (струдники кафедр, деканаты, учебный центр). Как показывает практика, большинство обращений в деканаты носит рутинный, справочный характер, что вынуждает сотрудников ежедневно работать в режиме справочника, приводя к перегрузкам и выгоранию. Кроме того, самим сотрудникам требуются данные из нормативных актов, которые невозможно удерживать в памяти, что также ведет к потере рабочего времени на ручной поиск.[2]

2. Отсутствие инструментов превентивного анализа академических трудностей. Традиционная модель контроля успеваемости носит констатирующий характер: проблема фиксируется только тогда, когда студент уже получил задолженность или не сдал экзамен. Отчисление студента несет для вуза прямые финансовые потери и снижает ключевые показатели эффективности. Раннее выявление потенциальных рисков позволило бы администрации применять превентивные меры: назначать дополнительные консультации, привлекать тьюторов или проводить профилактические беседы.

Задачи анализа успеваемости активно исследуются, однако попытка прямого внедрения готовых решений сталкивается с рядом ограничений.

Зарубежные системы: обладают встроенными предиктивными модулями, однако их математические модели опираются на Болонскую систему, что делает прогнозы нерелевантными для российского специалитета с жестко фиксированными учебными планами

Отечественные КИС: успешно справляются с транзакционными задачами (учет контингента, ведомости), однако констатируют факты и не применяют методы машинного обучения для предсказания результатов на основе исторических данных.[3]

Таким образом, необходимо создание собственного сервиса, адаптированного под специфику отечественного высшего медицинского учреждения. Целевая аудитория системы включает авторизованных обучающихся и сотрудников вуза.

Сформулированы следующие функциональные и нефункциональные требования к системе:

1. Обеспечение авторизации пользователей с привязкой к ролевой модели КИС.
2. Использование исключительно внутренних данных вуза для исключения генерации недостоверных данных.
3. Масштабируемость и отказоустойчивость при пиковых нагрузках.
4. Информационная безопасность: изоляция контуров обработки персональных данных студентов и деперсонализация запросов к языковым моделям.
5. Кроссплатформенность клиентского приложения с использованием единой кодовой базы.

6. Аналитический модуль должен в фоновом режиме анализировать цифровой след студентов и формировать вероятностный прогноз успешной сдачи промежуточной аттестации.

7. Модуль информационной поддержки должен уметь вести диалог, извлекая ответы строго из загруженной базы документов вуза для исключения галлюцинаций.

Исходя из сформулированных требований и преодоления ограничений монолитных систем, в работе осуществлено использование гибкой микросервисной архитектуры. Принцип разделения ответственности обусловлен тем, что задачи интеллектуального текстового поиска (RAG) и предиктивного моделирования (ML) имеют различную математическую природу и требуют дифференцированных вычислительных ресурсов. Модульный подход позволяет разрабатывать и тестировать функциональные блоки, что значительно упрощает их сопровождение и обеспечивает высокий потенциал для гибкого развития и масштабирования системы в будущем.[4]

Спроектированная архитектура разделена на четыре основных уровня:

1. Слой хранения и управления данными: фундамент системы, обеспечивающий хранение и интеграцию структурированных и неструктурированных данных.

2. Интеграционный слой и маршрутизация: связующее звено, отвечающее за асинхронное взаимодействие компонентов и распределение нагрузки.

3. Слой интеллектуальных сервисов: вычислительное ядро системы, представленное независимыми микросервисами поиска (RAG) и аналитики (ML).

4. Клиентский слой: уровень пользовательских интерфейсов, обеспечивающий доступ к функциям ассистента.

Источники данных системы. [5] Интеллектуальный помощник использует данные, которые можно разделить на два типа:

1. Структурированные данные: оперативная информация из КИС «1С:Университет» (успеваемость, расписание, учебные планы), доступ к которой осуществляется через специально разработанные HTTP-сервисы (REST API).

2. Неструктурированные данные: корпус локальных нормативных актов, приказов и регламентов университета, представленный в форматах PDF, DOCX и HTML.

Слой хранения и управления данными. На данном уровне архитектуры осуществляется интеграция и персистентное хранение данных из вышеуказанных источников:

1. Векторная база данных (ChromaDB): используется для хранения многомерных представлений (эмбеддингов) неструктурированных документов, что обеспечивает работу механизма семантического поиска в RAG-ассистенте.

2. Реляционная база данных (PostgreSQL): выполняет роль служебного хранилища для ведения истории диалогов (сохранение контекста), управления профилями пользователей и кэширования частых поисковых запросов.[6]

Интеграционный слой и маршрутизация. Связующее звено, обеспечивающее асинхронное взаимодействие компонентов. API-шлюз принимает запросы от клиентских приложений и ставит их в очередь отказоустойчивого брокера сообщений (RabbitMQ). Брокер распределяет тяжелые вычислительные задачи (генерация текста или расчет ML-прогноза) между свободными воркерами, защищая систему от падения при пиковых нагрузках. Разработка API-шлюза микросервиса выполнена на базе современного веб-фреймворка FastAPI, который нативно поддерживает асинхронный ввод-вывод. Это предотвращает

блокировку потоков сервера в моменты пиковых студенческих нагрузок, что критически важно, учитывая, что время ответа тяжелой нейросети может достигать нескольких секунд.

Слой интеллектуальных сервисов. Вычислительное ядро, реализованное на языке Python и состоящее из двух независимых микросервисов:

1. Модуль информационной поддержки (RAG-сервис): выполняет семантический поиск по векторной базе, формирует расширенный промпт с контекстом и отправляет его в локально развернутую большую языковую модель (LLM) для генерации финального ответа.
2. Модуль предиктивной аналитики (ML-сервис): осуществляет регламентный забор структурированных данных из 1С, проводит инженерию признаков и прогоняет данные через обученную модель градиентного бустинга для расчета вероятности академической задолженности.[7]

Клиентский слой. Единый пользовательский интерфейс, абстрагирующий пользователя от сложности внутренних процессов. Разработка ведется с применением кроссплатформенной технологии Kotlin Multiplatform и интеграции мини-приложения на языке Go (MAX Mini App) для снижения порога входа обучающихся и обеспечения безопасной OAuth-авторизации.

Данная архитектура обеспечивает гибкость разработки: она позволяет независимо обновлять языковую модель в RAG-модуле или алгоритм прогнозирования, не останавливая работу всего приложения и не вмешиваясь в монолитный код системы 1С.

Качество и достоверность работы диалоговых систем критически зависят от репрезентативности используемых данных. [8] В рамках исследования был принят принцип полного отказа от открытых датасетов, так как они не отражают специфику промежуточной аттестации и прохождения практик в университете. Эмпирической базой выступают исключительно внутренние неструктурированные нормативные акты (приказы, регламенты) и динамические структурированные данные из системы учета ЮУГМУ.

Для обработки неструктурированного контента (документы в форматах HTML, PDF и DOCX) спроектирован специализированный программный модуль парсинга, осуществляющий асинхронный обход веб-страниц порталов вуза. С целью повышения семантической плотности текста применен метод DOM-фильтрации, алгоритм предварительно находит и удаляет (через метод decompose) все неинформативные теги, что исключает попадание в векторную базу навигационного шума и предотвращает смещение внимания языковой модели на случайные контакты или элементы меню.[9]

Процесс подготовки извлеченных текстов включает фрагментацию и нормализацию. Предобработанные документы передаются в алгоритм рекурсивного разбиения. Ключевым архитектурным решением на этапе векторизации стало внедрение механизма дедупликации на основе криптографического хеширования (алгоритм SHA-256). Алгоритм вычисляет уникальный хеш каждого нормализованного фрагмента (чанка); векторизации подлежат только уникальные фрагменты. Это решает проблему дублирования данных, экономит память в векторной СУБД и предотвращает смещение фокуса LLM из-за многократного повторения одного факта.

Связующим звеном архитектуры генерации, дополненной поиском (RAG), выступает векторная база данных. Сравнительный анализ популярных решений показал, что использование облачного сервиса недопустимо политикой информационной безопасности вуза, а локальная база FAISS (Meta) обладает ограничениями в части персистентности и фильтрации метаданных. Для практической реализации выбрана встраиваемая СУБД

ChromaDB, оптимизированная для локальной работы, поддерживающая персистентное хранение и HNSW-поиск.[10]

Требования Федерального закона РФ № 152-ФЗ «О персональных данных» запрещают передачу контекста на серверы внешних провайдеров. В связи с этим вычислительное ядро развернуто локально на платформе Ollama с использованием 4-битного квантования весов, что позволило кардинально снизить требования к видеопамяти графических ускорителей. Как будет показано ниже в описании экспериментов, обусловивших выбор компонентов системы, будет выбрана модель Qwen3.5-4B в качестве генеративного ядра, обеспечивающая баланс между требованиями к аппаратному обеспечению (~3 ГБ VRAM) и высоким качеством генерации структурированных ответов на русском языке.

Качество финального ответа интеллектуального помощника зависит от гиперпараметров конвейера обработки данных. Была разработана серия экспериментальных стендов для эмпирической оценки и настройки каждого изолированного компонента системы.[11]

Исследовалось влияние размера текстового блока на семантическую целостность. На валидационном датасете были сгенерированы три конфигурации: мелкая (300 токенов, перекрытие 50), базовая (600 токенов, перекрытие 120) и крупная нарезка (1200 токенов, перекрытие 200). Эксперимент показал, что база мелкой нарезки (675 фрагментов) вырывает предложения из контекста, теряя логические связи регламентов, в то время как база крупной нарезки (159 фрагментов) приводит к размытию семантики и переполнению оперативной памяти. Базовая конфигурация (сгенерировавшая 331 фрагмент) была выбрана как лучшая, поскольку она вмещает целостный смысловой абзац документа, не перегружая контекстное окно языковой модели.[12]

Сравнивалась семантическая точность перевода текста в многомерный вектор на примере трех моделей: англоязычной nomic-embed-text (768 измерений), Multilingual-E5-large (1024 измерения) и qwen3-embedding:4b (1536 измерений). Оценка проводилась методом similarity_search_with_score (фреймворк LangChain). Выбор данного метода обусловлен необходимостью получения не только извлеченного текста, но и количественной метрики релевантности. Наличие числового значения позволило объективно сравнить семантическую точность различных моделей-энкодеров на едином тестовом наборе данных. В качестве данных использовались 30 страниц с сайта и 20 нормативных документов с различным наполнением. Англоязычная модель (nomic-embed-text) продемонстрировала высокую скорость, но потерпела полный семантический провал (поиск вернул документ о столовой вместо регламента пересдач), что доказывает не применимость моделей без предварительного обучения на русскоязычных корпусах. В свою очередь мультиязычная модель (Multilingual-E5-large) хоть и продемонстрировала высокую семантическую точность и высокую скорость поиска (0.0815 сек), но при этом не дошла до результатов qwen3, при более высоком потреблении ресурсов. Модель qwen3-embedding:4b показала высокую скорость поиска (0.0743 сек) и корректно извлекла релевантный нормативный акт, обеспечив баланс между качеством понимания контекста и скоростью работы в локальном контуре.

Важнейшим гиперпараметром является top-k, определяющий количество извлекаемых фрагментов, передаваемых в языковую модель. Замеры показали, что при минимальном значении (k=1, длина контекста 148 символов) модель фокусируется на частных случаях из-за недостатка данных. При избыточном значении (k=10, контекст более 5200 символов) зафиксирована галлюцинация: из-за попадания в промпт документов низкого ранга нейросеть

смешала понятия академического отпуска и задолженности. Параметр k=3 (длина контекста ~1338 символов) выбран как рабочий, гарантирующий исчерпывающий ответ без риска зашумления промпта.

Кроме того, плотные векторные представления часто уступают классическим лексическим алгоритмам при поиске точных совпадений (специфических аббревиатур или номеров справок, например «095/у»). Был спроектирован стенд гибридного ансамблевого поиска (Ensemble Retriever), объединяющий векторный поиск и статистический алгоритм BM25 (с весами 0.5/0.5). Гибридный поиск позволил нивелировать недостатки каждого метода в отдельности, максимизируя полноту извлекаемого контекста.

Также проводился эксперимент с алгоритмом HyDE для повышения точности поиска. Его суть в том, чтобы генерировать примерный ответ на вопрос пользователя и переводить его в более понятный язык для поиска в векторной базе. Предполагалось, что данный алгоритм повысит точность, однако он показал его избыточность при использовании мощной модели qwen3 и был исключен из базовой архитектуры для снижения общей задержки.

Для предотвращения заполнения контекста дублирующимися абзацами классический косинусный поиск (Similarity Search) был сравнен с алгоритмом максимальной маржинальной релевантности (MMR). Суть эксперимента заключалась в подаче запроса с комплексным вопросом, объединяющего в себе два вопроса (Как получить стипендию и каков ее размер?) к базе содержащей семантические дубликаты. Эксперимент доказал способность алгоритма MMR гарантировать разнообразие контекста без потери производительности (скорость поиска 0.0643 сек).

Процесс обработки запроса в реальном времени реализован в виде гибридного пайплайна: система извлекает K=3 релевантных фрагментов из векторной БД и в случае запроса студента параллельно выполняет REST API-запрос к КИС «1С: Университет» для извлечения оперативного цифрового профиля студента (оценки, задолженности). Данные программно объединяются в единый системный промпт, что обеспечивает абсолютную персонализацию: модель опирается на актуальную выгрузку из 1С, формируя точные административные рекомендации.

Модуль предиктивной аналитики предназначен для перехода от констатирующего контроля успеваемости к раннему выявлению студентов с высоким риском отчисления. Решение данной задачи требует перевода транзакционных данных КИС в структурированный вектор признаков и подбора оптимального алгоритма бинарной классификации.

Выгрузка исторических данных (около 500 000 уникальных записей) реализована через специализированный программный модуль на встроенном языке 1С. Динамический набор данных формируется путем запросов к регистрам: «Посещаемость» (количество присутствий и пропусков), «Результаты вступительных испытаний» (средний балл ЕГЭ) и «Аттестация» (результаты экзаменов, историческое количество пересдач). Каждая строка итогового датасета представляет собой вектор состояния студента по одной учебной дисциплине.

На этапе математической предобработки выполнена строгая деперсонализация и форматирование числовых переменных. Ключевым шагом инженерии стала бинаризация целевой переменной: исходная мультиклассовая задача (оценки от «Отлично» до «Неявка») была сведена к двум классам. В Класс 1 («Группа риска») объединены оценки «Неудовлетворительно», «Не зачтено» и «Неявка», а в Класс 0 («Успешная сдача») — все

положительные результаты. Это позволило сфокусировать алгоритмы на поиске паттернов академической задолженности.

Экспериментальный стенд сравнивал три класса алгоритмов: логистическую регрессию, random forest и градиентный бустинг (CatBoost). Логистическая регрессия была выбрана в качестве baseline, так как данный алгоритм позволяет оценить, являются ли классы линейно разделимыми. Random forest в свою очередь позволяет решить проблему нелинейности данных, но не способен нативно обрабатывать категориальные признаки, требуя их обязательного предварительного кодирования. Градиентный бустинг в частности Catboost способен использовать категориальные признаки. Данные разделены на Train/Test в пропорции 80/20 со стратификацией stratify=y для сохранения доли студентов группы риска.

В ходе экспериментов классические алгоритмы (логистическая регрессия и random forest) потребовали применения прямого кодирования для текстового признака «Дисциплина» и стандартизации StandardScaler. Это привело к избыточному разрастанию размерности матрицы и фрагментации данных. Алгоритм CatBoost принимал строковые названия, напрямую самостоятельно осуществляя целевое кодирование, что сэкономило ресурсы и предотвратило утечку данных.

Тестирование на отложенной выборке доказало превосходство градиентного бустинга. Линейная модель показала низкую способность к обобщению (F1-Score = 0.5647), так как влияние пропусков на итоговую оценку имеет сложную нелинейную зависимость от сложности предмета. Случайный лес показал приемлемую точность, но уступил по ключевой метрике полноты. CatBoost продемонстрировал наивысшие показатели: Accuracy = 0.9530, Precision = 0.8611 и Recall = 0.8155, успешно выявляя более 81% студентов на грани отчисления.

Для математического доказательства отсутствия переобучения (Overfitting) проведен эксперимент со стратифицированной K-Fold кросс-валидацией (5 фолдов). Результаты зафиксировали среднее значение F1-Score на уровне 0.8312 со стандартным отклонением 0.014. Низкая дисперсия ошибки (<1.5% между фолдами) статистически доказывает, что модель CatBoost находит устойчивые глобальные паттерны, а не заучивает данные наизусть.

Оптимизация выбранной модели CatBoost проводилась с учетом бизнес-логики деканата. Академические данные имеют структурный дисбаланс (доля отчисляемых составляет около 20%). Для устранения тривиального предсказания мажоритарного класса внедрен механизм алгоритмического взвешивания (auto_class_weights='Balanced'), штрафующий функцию потерь за ошибку на миноритарном классе, что исключило необходимость использования зашумляющих синтетических данных.

Методом поиска по сетке (Grid Search) выбраны гиперпараметры, предотвращающие переобучение глубоких деревьев: iterations=200, depth=6, learning_rate=0.05.

Учитывая, что ложноотрицательное срабатывание (пропуск потенциального должника) ведет к отчислению и потере финансирования вузом, был проведен сдвиг порога классификации (Threshold). Снижение порога со стандартного 0.50 до 0.30 алгоритмически перевело модель в «пессимистичный» режим, что снизило Precision до 0.62, но максимизировало выявляемость группы риска (Recall) до 0.94.

Для обеспечения прозрачности системы проведен анализ важности признаков встроенными методами CatBoost. Установлена следующая иерархия предикторов академической неуспеваемости:

1. Процент пропусков занятий (~45%). Сильнейший предиктор: риск экспоненциально возрастает при превышении 15-20% пропущенных часов.
2. Учебная дисциплина (~25%). Выявлены дисциплины-маркеры («Биохимия», «Анатомия»), задолженность по которым резко повышает вероятность отчисления по сравнению с факультативами.
3. Историческое количество пересдач (~15%). Кумулятивный индикатор системных проблем обучающегося.
4. Средний балл ЕГЭ (~10%). Высокая предсказательная сила на первых курсах с постепенным затуханием на старших.

В рамках исследования решена проблема фрагментированности цифровых сервисов медицинского университета и дефицита предиктивных аналитических инструментов. Разработанная микросервисная архитектура интеллектуального помощника обеспечивает бесшовную и безопасную работу с персональными данными в изолированном контуре вуза.

Экспериментально подтверждена эффективность гибридного RAG-конвейера (на базе ChromaDB и LLM Qwen3.5-4B), исключающего проблему галлюцинаций за счет строгой опоры на локальные нормативные акты. Реализованный модуль машинного обучения (CatBoost) доказал свою высокую предсказательную способность (F1-Score: 0.8312), а проведенная оптимизация порога классификации позволила системе выявлять до 94% студентов группы риска на ранних стадиях. Внедрение архитектуры обеспечит существенное снижение административной нагрузки на сотрудников деканатов и создаст технологическую основу для превентивной работы по сохранению контингента обучающихся.

Список литературы

1. Федеральный закон от 27 июля 2006 г. № 152-ФЗ «О персональных данных» (с изм. и доп.). — Текст: электронный— URL: https://www.consultant.ru/document/cons_doc_LAW_61801/ (дата обращения: 12.04.2026).
2. Мюллер, Андреас, Гвидо, Сара. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. : Пер. с англ. — СПб. : ООО "Альфа-книга", 2017. - 480 с.: ил. - Парал. тит. англ. ISBN 978-5-9908910-8-1 (рус.)
3. Радченко, М. Г. 1С:Предприятие 8.3. Практическое пособие разработчика / М. Г. Радченко, Е. Ю. Хрусталева. — Москва : 1С-Паблишинг, 2013. — 926 с. Текст : электронный— URL: https://online.1c.ru/books/book/17628399 (дата обращения: 12.04.2026).
4. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах. — Москва : ДМК Пресс, 2015. — 400 с.
5. CatBoost - open-source gradient boosting library : official documentation. — Текст : электронный // CatBoost : [сайт]. — URL: https://catboost.ai/en/docs/ (дата обращения: 10.04.2026).
6. Chroma: the AI-native open-source embedding database. — Текст : электронный // Chroma : [сайт]. — URL: https://docs.trychroma.com/ (дата обращения: 11.04.2026).
7. FastAPI framework, high performance, easy to learn, fast to code, ready for production. — Текст : электронный // FastAPI : [сайт]. — URL: https://fastapi.tiangolo.com/ (дата обращения: 12.04.2026).

8. LangChain Documentation: Building applications with LLMs through composability. — Текст : электронный // LangChain : [сайт]. — URL: https://python.langchain.com/docs/get_started/introduction (дата обращения: 14.04.2026).
9. Lewis, P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis, E. Perez, A. Piktus // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — P. 9459-9474. — Текст : непосредственный.
10. Богатырев, С. Ю. Машинное обучение в финансах : учебник / С. Ю. Богатырев, А. А. Помулев, А. В. Затевахина ; под редакцией С. Ю. Богатырева. — Москва : Прометей, 2024. — 224 с. — ISBN 978-5-00172-572-5. — Текст : электронный
11. Митяков, Е. С. Искусственный интеллект и машинное обучение : учебное пособие для вузов / Е. С. Митяков, А. Г. Шмелева, А. И. Ладынин. — 2-е изд., стер. — Санкт-Петербург : Лань, 2026. — 252 с. — ISBN 978-5-507-51198-3. — Текст : электронный
12. Филиппова, А. С. Курс лекций по дисциплине «Цифровые технологии в научно-исследовательской и управленческой деятельности» : учебно-методическое пособие / А. С. Филиппова, Э. И. Дяминова, Ф. З. Забихуллин. — Уфа : БГПУ имени М. Акмуллы, 2025. — 114 с. — ISBN 978-5-00251-024-5. — Текст : электронный

References

1. Federal Law No. 152-FZ of 27 July 2006 ‘On Personal Data’ (as amended and supplemented). — Text: electronic— URL: https://www.consultant.ru/document/cons_doc_LAW_61801/ (accessed: 12 April 2026).
2. Müller, Andreas, Guido, Sarah. Introduction to Machine Learning with Python. A Guide for Data Professionals. Translated from English. — St Petersburg: Alfa-Kniga LLC, 2017. — p.480: ill. — Parallel title in English. ISBN 978-5-9908910-8-1 (Russian)
3. Radchenko, M. G. 1C:Enterprise 8.3. A Practical Guide for Developers / M. G. Radchenko, E. Yu. Khrustaleva. — Moscow: 1C-Publishing, 2013. — p. 926 Text: electronic— URL: https://online.1c.ru/books/book/17628399 (accessed: 12 April 2026).
4. Flah, P. Machine Learning. The Science and Art of Building Algorithms that Extract Knowledge from Data / P. Flah. — Moscow: DMK Press, 2015. — p. 400
5. CatBoost - open-source gradient boosting library: official documentation. — Text: electronic // CatBoost: [website]. — URL: https://catboost.ai/en/docs/ (accessed: 10 April 2026).
6. Chroma: the AI-native open-source embedding database. — Text : electronic // Chroma : [website]. — URL: https://docs.trychroma.com/ (accessed: 11 April 2026).
7. FastAPI framework, high performance, easy to learn, fast to code, ready for production. — Text : electronic // FastAPI : [website]. — URL: https://fastapi.tiangolo.com/ (accessed: 12 April 2026).
8. LangChain Documentation: Building applications with LLMs through composability. — Text: electronic // LangChain: [website]. — URL: https://python.langchain.com/docs/get_started/introduction (accessed: 14 April 2026).
9. Lewis, P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis, E. Perez, A. Piktus // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — pp. 9459–9474. — Text: direct.

10. Bogatyrev, S. Yu. Machine Learning in Finance: A Textbook / S. Yu. Bogatyrev, A. A. Pomulev, A. V. Zatevakina; edited by S. Yu. Bogatyrev. — Moscow: Prometheus, 2024. — p. 224— ISBN 978-5-00172-572-5. — Text: electronic
11. Mityakov, E. S. Artificial Intelligence and Machine Learning: A Textbook for Universities / E. S. Mityakov, A. G. Shmeleva, A. I. Ladynin. — 2nd ed., rev. — Saint Petersburg: Lan, 2026. — p.252 — ISBN 978-5-507-51198-3. — Text: electronic
12. Filippova, A. S. Lecture Course on the Subject ‘Digital Technologies in Research and Management Activities’: Teaching Guide / A. S. Filippova, E. I. Dyaminova, F. Z. Zabikhullin. — Ufa: M. Akmulla Ufa State Pedagogical University, 2025. — p. 114— ISBN 978-5-00251-024-5. — Text: electronic