МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра социальной информатики

САЙТ-СИСТЕМА ОБРАБОТКИ ДАННЫХ. DATA ANALYSIS В ВЕБ-ПРИЛОЖЕНИЯХ

(автореферат бакалаврской работы)

Студента 4 курса 451 группы
направления 09.03.03 - Прикладная информатика
профиль Прикладная информатика в социологии
Социологического факультета
Кашкина Владислава Дмитриевича

Научный руководитель, старший
преподаватель
кафедры социальной информатики _____ Ю.А. Седавкина
подпись, дата

Зав. кафедрой
кандидат социологических наук, доцент _____ И.Г. Малинский
подпись, дата

Саратов 2025

ВВЕДЕНИЕ

Актуальность выпускной квалификационной работы обусловлена постепенным переходом к «цифровой экономике», которая сопровождается резким ростом объёма общественно значимых данных, что требует современных инструментов анализа и визуализации.

В России в рамках федерального проекта «Государственная статистика» создаётся единая Цифровая аналитическая платформа Росстата (ГИС ЦАП) для сбора и обработки статистических данных. К 2030 году планируется автоматизировать импорт всех статистических данных в Росстат по единому стандарту, что позволит повысить их качество и оперативность аналитики. Параллельно растёт доступность открытых данных: по оценкам OECD, усилия по публикации открытых данных в последние годы приобретают всё возрастающую роль в инновациях и государственном управлении. В совокупности это определяет **актуальность** разработки интерактивных веб-инструментов для обработки табличных данных (CSV, Excel и др.), которые обеспечивают гибкий доступ к аналитическим функциям в браузере и отвечают потребностям прикладной аналитики (включая социальную и экономическую) при минимальных требованиях к пользовательскому «железу» и лицензиям.

Современная наука и практика уже демонстрируют возможность полного цикла анализа данных на стороне клиента. Так, Karampakakis и соавт. описали веб-приложение для умного города, где данные об окружающей среде динамически обрабатываются, анализируются и визуализируются прямо в браузере. Появились специализированные инструменты и библиотеки: например, SOCRAT – это масштабируемый веб-фреймворк для визуальной аналитики, реализованный на модульной архитектуре, который интегрирует компоненты импорта данных, интерактивной визуализации и статистических вычислений. Библиотеки визуализации, такие как Chart.js, позволяют создавать адаптивные и красивые диаграммы во множестве стандартных форматов (линейные, столбчатые, точечные, круговые и др.), а средства обработки данных, например Danfo.js, предлагают структуры данных для манипуляции табличными

наборами в стиле Python Pandas. Таким образом, в научном сообществе признаётся растущий потенциал веб-технологий для интерактивного анализа данных на клиенте и сервере, позволяющий объединять вычисления и визуализацию в единой среде.

Целью настоящей работы является **разработка и апробация веб-приложения** на базе Python-фреймворка Tornado, реализующего основные методы анализа табличных данных. Приложение должно обеспечивать загрузку и хранение наборов данных (CSV/Excel), автоматический расчёт описательной статистики, построение корреляций и линейных регрессий, визуализацию результатов и базовые инструменты машинного обучения. Особое внимание уделяется простоте пользовательского интерфейса и независимости от проприетарных платформ и лицензий.

Для достижения цели сформулированы следующие **задачи исследования**:

1. Анализ существующих решений и архитектур: обзор современных подходов и инструментов веб-анализа данных (как клиентских, так и серверных), включая веб-фреймворки, библиотеки визуализации и вычислений (например, SOCRAT, Observable, Chart.js, Danfo.js и др.), а также их сравнение с традиционными средствами (Pandas, NumPy).

2. Проектирование архитектуры приложения**:** определение структуры веб-системы, включая выбор компонентов для хранения данных, вычислений и визуализации; разработка схемы взаимодействия «клиент–сервер» с использованием Tornado на сервере и JavaScript/HTML5 на стороне клиента.

3. Реализация ключевых модулей: разработка модулей описательной статистики, корреляционно-регрессионного анализа, визуализации (диаграмм, графиков) и базовых ML-методов (например, k-средних, kNN); обеспечение загрузки и выгрузки данных в привычных форматах.

4. Тестирование и валидация: апробация веб-приложения на реальных данных из открытых источников (например, статистических наборов Росстата) и на тестовых таблицах. Проверка корректности вычислений (сравнение с

библиотеками Python/Pandas) и оценка производительности при разных объёмах данных.

5. Документирование результатов: оформление руководства пользователя и технической документации, анализ сильных и слабых сторон созданного приложения, рекомендации по его дальнейшему развитию и масштабированию.

Объектом исследования является веб-приложение для анализа табличных данных. **Предметом** выступают методы аналитической обработки данных и их интерактивного представления в браузере. Теоретическая база включает концепции визуального восприятия информации и теорию обработки данных, а также методологии веб-прототипирования и человеко-компьютерного взаимодействия. В работе применяются статистические методы (описательная статистика, корреляционный и регрессионный анализ), методы машинного обучения, сравнительный анализ программных решений и разработки интерфейсов.

В качестве **эмпирической базы** используются открытые статистические наборы (в том числе от Росстата и международных организаций) и специально подготовленные CSV/Excel-файлы. Эти данные позволяют протестировать корректность импорта, обработки и экспорта результатов через веб-интерфейс. Разработанное приложение поддерживает загрузку таблиц, их предварительную обработку (например, нормализацию, заполнение пропусков) и сохранение итоговых отчётов в удобных форматах для последующего использования исследователями и студентами.

Практическая значимость работы заключается в создании доступного веб-инструмента для анализа данных, который не требует установки специального ПО и не зависит от лицензионных ограничений. Такие решения повышают вовлечённость широкого круга пользователей (исследователей, аналитиков, студентов) в обработку данных и могут масштабироваться в сетевых приложениях без существенных затрат на инфраструктуру. Демонстрация работы вычислительных модулей в браузере подчёркивает образовательную ценность проекта: студенты и социологи получают возможность проводить

аналитическую обработку данных в знакомой среде, концентрируясь на интерпретации результатов.

Структура выпускной работы традиционна: она включает введение, три главы (аналитическую, проектную и экспериментальную части), заключение, список использованных источников и приложение. Во введении обоснованы актуальность, цель и задачи исследования; в первой главе рассматриваются теоретические основы анализа данных и обзор существующих веб-инструментов; во второй – описывается архитектура и реализация разработанного приложения; в третьей приводится описание экспериментов и анализ результатов. Заключение содержит основные выводы и рекомендации.

Основное содержание работы

В первом разделе «Аналитическая часть: теоретические основы и обзор» раскрываются предпосылки и мотивация создания веб-систем для интерактивного анализа табличных данных в браузере, начиная с описания экспоненциального роста объёмов данных и потребности в демократизации доступа к аналитическим инструментам без установки сложного ПО и дорогостоящих лицензий, что обосновывается исследованиями по «Большим данным» и их влиянию на развитие аналитики в различных областях . Далее рассматривается наличие открытых статистических ресурсов (например, порталы Росстата и международные источники), стимулирующих запросы исследователей и практиков на быструю предварительную обработку и визуализацию данных прямо в браузере.

Появляются три основных подхода к организации вычислений: полностью серверный, полностью клиентский и гибридный. Полностью серверный подход обеспечивает возможность обработки очень больших массивов и выполнения вычислительно тяжёлых задач с использованием экосистемы Python (Pandas, NumPy, SciPy, scikit-learn), но он сопровождается снижением отзывчивости интерфейса из-за сетевых задержек и может требовать значительных серверных ресурсов. Клиентский подход, основанный на современных браузерных движках JavaScript и зрелых библиотеках (Danfo.js для

DataFrame-подобных операций, simple-statistics.js/jStat.js для статистических расчётов, Chart.js, D3.js, Plotly.js для визуализации), позволяет мгновенно реагировать на действия пользователя и проводить описательную статистику, базовые инференциальные тесты и простые ML-алгоритмы (k-Means, kNN, простая линейная регрессия) локально в браузере. Однако браузер ограничен по памяти и выполняет код в основном в однопоточном режиме, что требует использования Web Workers и WebAssembly для фоновых и более производительных вычислений, но не снимает полностью проблемы при очень больших объёмах данных и усложняет архитектуру приложения. Гибридная модель признана наиболее сбалансированной: небольшие и средние по объёму операции (загрузка через File API, первичная валидация, описательная статистика, визуализация, простые тесты и ML) выполняются на клиенте для интерактивности, а более тяжёлые или большие задачи перенаправляются на сервер через REST/JSON API или WebSocket в асинхронном фреймворке Tornado на Python, что позволяет задействовать библиотеки Pandas/NumPy/SciPy/scikit-learn без блокировки UI и с гарантиями корректности результатов .

Далее подробно анализируются технологии клиентской части: описывается использование File API для чтения CSV/Excel (с учётом кодировок и форматов), валидация данных (проверка типов столбцов, выявление пропусков, обработка «грязных» данных: удаление или замена медианой/средним с явным уведомлением пользователя). Для описательной статистики применяются функции jStat/simple-statistics.js: подсчёт среднего, медианы, моды, дисперсии, стандартного отклонения, асимметрии, эксцесса и квантилей. Визуализация распределений через гистограммы и boxplot реализуется с помощью Chart.js (включая плагины для boxplot и histogram), D3.js или Plotly.js с интерактивными подсказками, зумом и фильтрацией «на лету». Для инференциальной статистики в браузере реализуются t-тесты (одновыборочный, независимые выборки, парный) и непараметрические тесты (U-тест Манна–Уитни, Wilcoxon), включая проверку предпосылок о нормальности (с использованием простых тестов Шапиро–Уилка или бутстрэп-

методов). При превышении объёмов, когда клиентская обработка становится неэффективной, запросы направляются на сервер, где SciPy выполняет расчёты быстрее и точнее. В корреляционно-регрессионном анализе простая линейная регрессия может выполняться локально при малых объёмах, оценка коэффициентов методом наименьших квадратов и визуализация линии регрессии, тогда как множественная регрессия и большие выборки обрабатываются на сервере.

В части ML подробно обсуждается реализация k-Means и kNN в клиенте: алгоритм кластеризации Lloyd выполняется в Web Worker при небольших наборах (до нескольких тысяч точек), с нормализацией (стандартная или min-max) и визуализацией кластеров через scatter (для многомерных данных – проекция через PCA, выполняемая на клиенте при небольших объёмах или на сервере при больших) . kNN применяется для классификации/регрессии на небольших обучающих выборках в браузере; при больших объёмах или запросах на оценку качества (силуэтный анализ, метрики точности) переключение на сервер. Пороговые величины определяются экспериментальными замерами производительности: описательная статистика остаётся интерактивной до десятков тысяч строк, t-тесты и регрессия – до нескольких тысяч, кластеризация – до трёх-пяти тысяч точек. Архитектурные и технологические аспекты рассматривают построение SPA: статические файлы (HTML, CSS, JS) подаются Tornado, а JavaScript инициализирует UI-компоненты (например, Handsontable для таблиц), связывая действия пользователя с модулями обработки или AJAX-запросами к REST/JSON API сервера, ответы от которого приходят в JSON и отображаются без перезагрузки страницы, обеспечивая отзывчивость интерфейса и экономию ресурсов. Для временного хранения сессий и данных на клиенте используется localStorage/IndexedDB, что позволяет сохранить состояние между перезагрузками без повторной загрузки больших файлов. При этом обсуждаются вопросы безопасности: HTTPS для шифрования трафика, CORS, авторизация (JWT), защита от CSRF и XSS, опциональная клиентская обработка конфиденциальных данных без передачи на сервер или шифрование

при передаче. Оптимизация производительности достигается использованием Web Workers для фоновых вычислений, WebAssembly для критически интенсивных операций, CDN для доставки библиотек, ленивой загрузкой модулей, агрегацией данных перед визуализацией больших наборов и дебаунсингом событий интерфейса.

Особое внимание уделяется сравнительному анализу клиентских и серверных реализаций одинаковых алгоритмов: корректность проверяется сопоставлением результатов с эталонными расчётами в Python/Pandas/SciPy, производительность замеряется для разных объёмов данных, что позволяет обосновать пороги переключения на сервер и стратегии кеширования результатов. Упомянуто использование инструментов тестирования: юнит-тесты клиентских функций через Jest или Mocha с Chai, интеграционные E2E-тесты с Playwright, нагрузочное тестирование Tornado API с учётом горизонтального масштабирования и очередей задач (Celery) для длительных вычислений. Также описаны принципы логирования и отладки: try…catch в клиентских модулях для «грейсфул деградации», логирование ошибок в консоль и на сервере Tornado, использование DevTools и Playwright Inspector/Trace Viewer для выявления асинхронных проблем.

Обзор литературных источников охватывает труды по дизайну взаимодействия (Cooper et al. «About Face»), адаптивному веб-дизайну (Gustafson), JavaScript (Flanagan, Freeman & Robson, Eloquent JavaScript), REST/WebSocket (Fielding, Richardson & Ruby, Fette & Melnikov), безопасность (Rescorla TLS, Same-origin policy, JWT), Web Storage (W3C), Web Workers/WebAssembly, статистику и ML (Hacking, James et al. «An Introduction to Statistical Learning», Hogg et al. «Probability and Statistical Inference», Siegel & Castellan, Efron & Tibshirani «Bootstrap»), визуализацию (Bostock D3.js, Chart.js, Plotly), библиотеки для JS-анализа данных (Danfo.js), а также исследования по Big Data и демократизации данных (Hilbert, Manyika, Saha & Srivastava), демонстрирующие, что выбранные методы и технологии обоснованы

современной практикой и позволяют обеспечить интерактивный, доступный и точный анализ данных в веб-среде.

Выводы по первому разделу: подтверждена высокая актуальность разработки веб-приложения для интерактивного анализа табличных данных в браузере с гибридной архитектурой, сочетающей клиентские вычисления с возможностью переноса тяжёлых задач на сервер; обоснован выбор стека (Tornado на сервере и JavaScript-библиотеки Danfo.js, simple-statistics.js/jStat.js, Chart.js/D3.js/Plotly.js на клиенте, Web Workers/WebAssembly для оптимизации); определены ключевые методы статистики (описательная, инференциальная) и базовые ML-алгоритмы (k-Means, kNN, линейная регрессия) с учётом их применимости и порогов объёма данных; сформулированы требования к обработке «грязных» данных, безопасности, производительности, хранению сессий и тестированию (юнит-, интеграционные и нагрузочные тесты), что создаёт прочную основу для проектирования и реализации веб-системы.

Во втором разделе «Проектная часть: проектирование и реализация веб-системы анализа данных» рассматриваются все этапы перехода от сформулированных требований к конкретным техническим решениям и реализациям, отражённые в ВКР. Помимо описания целевой аудитории и сценариев использования, подробно анализируются функциональные и нефункциональные требования, прорабатывается выбор стека технологий и архитектурной модели, описываются детали реализации клиентских и серверных модулей, взаимодействия между ними, а также стратегия тестирования и сопровождения системы.

Сначала раскрывается сбор и формализация требований, где подчёркивается необходимость учитывать реальные потребности исследователей, студентов и малых предприятий, не обладающих глубокими навыками программирования и не желающих устанавливать дополнительное ПО, но нуждающихся в быстром разведочном анализе табличных данных и базовом статистическом моделировании. На этой основе определяются функциональные требования: импорт таблиц из Excel (.xlsx, .xls) и CSV,

интерактивное редактирование данных в браузере с компонентом, подобным электронной таблице, экспорт обратно в Excel, сохранение состояния (локальное сохранение в браузере) для возобновления работы. Аналитические функции должны включать вычисление описательной статистики (количество наблюдений, среднее, медиана, мода, стандартное отклонение, минимум, максимум и т.д.), выполнение базовых статистических тестов (t-тест Стьюдента, непараметрический U-тест Манна–Уитни), реализацию алгоритмов машинного обучения (кластеризация k-means, классификация k-ближайших соседей, простая линейная регрессия) и построение интерактивных визуализаций (гистограммы, scatter plot, boxplot).

Нефункциональные требования фокусируются на высокой интерактивности и отзывчивости интерфейса, доступности без установки ПО (достаточно современного браузера), автономности работы офлайн после первоначальной загрузки, строгой конфиденциальности данных (анализ происходит исключительно на стороне клиента без передачи на внешние серверы). Дополнительно определены технологические и эксплуатационные ограничения: использование исключительно открытого ПО (open source) и минимальная серверная инфраструктура, оптимально — однократная отдача статических файлов с минимальной нагрузкой на бэкенд. Эти требования побуждают рассмотреть преимущественно клиентскую архитектуру с возможностью расширения серверной части лишь при необходимости обработки очень больших наборов или сложных моделей.

Далее идёт обоснование выбора стека технологий. С точки зрения бэкенда показан выбор Tornado на Python как лёгкого асинхронного фреймворка, обеспечивающего возможность в будущем добавлять API-эндпоинты для тяжёлых вычислений (Pandas/NumPy/SciPy/scikit-learn) без избыточного каркаса. Выбирается чистый JavaScript в браузере для реализации основной логики: File API (SheetJS для чтения/записи Excel, Papa Parse для CSV), библиотеки обработки данных (Danfo.js или собственные обёртки вокруг jStat/simple-statistics.js), визуализации (Chart.js, D3.js, Plotly.js), Web Workers/WebAssembly

для фоновых вычислений, Handsontable-подобный компонент для интерактивной таблицы, jsPDF для экспорта отчётов в PDF и прочие инструменты. Поясняется, что хотя Tornado позволяет выполнять асинхронные запросы и в будущем масштабировать систему, в текущей реализации сервер отвечает лишь за отдачу статических файлов, а вся аналитика выполняется на клиенте, что полностью соответствует требованиям автономности и конфиденциальности.

Затем описывается архитектурная модель SPA: при первоначальном заходе пользователь получает HTML/CSS/JS от Tornado, после чего JavaScript инициализирует UI — таблицу, панели управления аналитическими модулями, визуализационные контейнеры и т. п. Взаимодействие с сервером сведено к статической доставке ресурсов, но при необходимости расширения бэкенда предусмотрена схема REST/JSON API (например, для k-means или регрессии при переработке очень больших объёмов) или WebSocket для уведомлений о статусе долгих задач. При загрузке данных File API проверяет формат, кодировку и наличие пропусков: строки с критичным числом пропусков удаляются или заполняются медианой/средним, с явным уведомлением пользователя, чтобы избежать некорректных результатов.

Реализация модулей на клиенте разбивается на логические части: data-loader (чтение и валидация через SheetJS/Papa Parse), data-processor (обёртки Danfo.js/jStat/simple-statistics.js для описательной и инференциальной статистики), ml-модуль (k-means и kNN через Web Worker с проверкой объёма данных и нормализацией), visualization (Chart.js/D3.js/Plotly.js для гистограмм, scatter, boxplot и тепловых карт), ui-controller (связывает события пользователя с обработчиками и при необходимости подготовкой запросов к серверу), error-handler (централизованная обработка ошибок и уведомления) и export-модуль (Excel через SheetJS, PDF через jsPDF). Пороговые величины для запуска клиентских вычислений определены экспериментально: описательная статистика остаётся интерактивной до десятков тысяч строк, t-тесты и простая регрессия — до нескольких тысяч, k-means — до трёх-пяти тысяч точек; при

превышении этих значений предлагается отправить данные на сервер для расчётов.

Особое внимание уделяется безопасности: весь трафик по HTTPS, настройка CORS, возможное использование JWT для аутентификации в многопользовательских сценариях, защита от CSRF/XSS; при строгом требовании конфиденциальности предусмотрена исключительно клиентская обработка чувствительных данных без передачи на сервер. Поясняется, что такая тонкая архитектура сервера (thin server) упрощает развёртывание и снижает требования к инфраструктуре, позволяя работать даже на простейшем статическом хостинге.

Стратегия тестирования охватывает юнит-тесты клиентских функций через Jest или Mocha с Chai для проверки корректности статистических расчётов и ML-алгоритмов, интеграционные E2E-тесты с Playwright для сценариев «загрузка данных — анализ — визуализация — экспорт», а также нагрузочное тестирование бэкенд-API Tornado при добавлении эндпоинтов тяжёлых вычислений, с проверкой горизонтального масштабирования и очередей задач (Celery) для длительных задач. Сравнение результатов клиентских вычислений с Python/Pandas/SciPy подтверждает точность реализованных алгоритмов. Для отладки используются логирование в браузере и на сервере, DevTools и Playwright Inspector/Trace Viewer для выявления асинхронных проблем.

Наконец, обсуждаются аспекты сопровождения и расширения: возможность добавления новых аналитических модулей на сервере, интеграция с облачными сервисами или базами данных при необходимости обработки больших объёмов, использование CDN для ускорения доставки клиентских библиотек, кеширование результатов часто выполняемых анализов, мониторинг и логирование серверных метрик через стандартные инструменты. Всё это обосновывается на базе требований и ограничений, выведенных в самом начале раздела и подтверждённых литературным обзором и экспериментальными замерами.

Выводы по второму разделу: подробно проанализированы и оформлены требования целевой аудитории и системы, оправдан выбор клиентской архитектуры с тонким сервером и возможностью расширения, описана реализация ключевых модулей загрузки, обработки, аналитики и визуализации, определены пороговые механизмы переключения на серверную обработку, реализованы меры по безопасности и автономности, разработана комплексная стратегия тестирования и отладки, а также учтены сценарии дальнейшего развития и масштабирования, что создаёт прочную и гибкую основу практической реализации веб-системы анализа данных.

В третьем разделе раскрывается проведение серии экспериментов и демонстраций, призванных подтвердить функциональность и корректность разработанных модулей веб-приложения при реальной работе с данными, а также определить границы применимости клиентской обработки и возможности расширения на сервер. Для этого было выбрано несколько репрезентативных наборов, включая социально-экономические показатели субъектов РФ (данные Росстата за 2023–2024 годы по разным регионам), что позволило смоделировать реальные сценарии анализа.

Сначала выполняется описательный анализ: данные загружаются через File API, проверяются формат и кодировка, очищаются «грязные» записи — например, строки с избыточным количеством пропусков удаляются или пропуски заменяются медианой или средним с явным уведомлением пользователя, чтобы не нарушать корректность вычислений. Далее с помощью клиентских JS-модулей подсчитываются ключевые характеристики: количество наблюдений, среднее, медиана, мода, дисперсия, стандартное отклонение, асимметрия, эксцесс и квантильные показатели для числовых столбцов. Визуализация распределений осуществляется интерактивно через Chart.js/D3.js/Plotly.js — строятся гистограммы, boxplot’ы, при необходимости выполняется агрегация или разбиение на бины непосредственно в браузере. Такое представление позволяет быстро выявлять выбросы и аномалии, сравнивать показатели между группами (например, уровни средней зарплаты по

регионам), а возможность зума и подсказок даёт глубокое понимание распределений.

Далее проводится инференциальный анализ: проверяется нормальность распределения (например, при небольших объёмах применяется тест Шапиро–Уилка или бутстрэп-методы на клиенте), затем выполняются t-тесты (одновыборочный, независимые выборки, парный) или непараметрические U-тесты Манна–Уитни/Wilcoxon при нарушении предпосылок. Для небольших выборок всё осуществляется в браузере, но при превышении экспериментально установленного порога клиент, обнаружив, что объём данных велик или требуется более точная проверка, посылает запрос на сервер, где SciPy выполняет расчёты с высокой точностью и скоростью. Результаты (p-value, доверительные интервалы разницы средних и т.д.) возвращаются в JSON и отображаются с визуализацией плотностей распределений и сравнительных boxplot'ов, что облегчает интерпретацию практической значимости обнаруженных различий.

Модели машинного обучения демонстрируются через кластеризацию k-Means: данные нормализуются (стандартный или min–max скейлинг) и передаются в Web Worker для выполнения итеративного алгоритма Lloyd; при небольшом объёме (до нескольких тысяч точек) кластеризация выполняется полностью на клиенте, при больших объёмах задачи направляются на серверную часть. Результаты кластеризации визуализируются через scatter-диаграммы с цветовой маркировкой по меткам кластеров, а при многомерности используется PCA-проекция: для малых наборов PCA может выполняться в браузере, для более крупных — на сервере. Кратко упоминается kNN для задач классификации/регрессии на небольших обучающих выборках, причём оценка качества моделей (силуэтный анализ для кластеров, метрики точности для kNN) инициирует отправку вычислений на сервер при превышении порогов.

Визуализация результатов экспериментов остаётся важным элементом: строятся гистограммы, scatter-графики, boxplot'ы, тепловые карты корреляций. Пользователь выбирает столбцы для осей, настраивает число бинов, включается

интерактивное масштабирование и подсказки. Для очень больших наборов применяются методы агрегации или предварительного вычисления на сервере, чтобы не перегружать браузер. Экспорт графиков в PNG/SVG осуществляется средствами библиотек непосредственно в интерфейсе.

Ключевой частью экспериментальной части является оценка производительности: замеры времени выполнения описательной статистики, t-тестов, регрессии, кластеризации на разных объёмах данных демонстрируют, что описательная статистика остается интерактивной до десятков тысяч строк, t-тесты и простая регрессия — до нескольких тысяч записей, k-Means — до трёх–пяти тысяч точек; выше этих порогов предпочтительнее перенести вычисления на сервер . Эти пороги установлены эмпирически через тесты в браузере с Web Workers и на сервере с Pandas/SciPy, что позволяет предложить пользователю автоматический выбор режима обработки. В документации фиксируются конкретные значения времени: например, среднее время описательной статистики для 10 000 строк составляет порядка секунды, для 50 000 — несколько секунд, тогда как на сервере аналогичная операция быстрее и не влияет на отзывчивость интерфейса.

В обсуждении результатов экспериментальной части подчёркивается, что клиентские вычисления дают высокую интерактивность и автономность, особенно для небольших и средних наборов, что важно для исследователей без доступа к мощным серверам. При этом отмечаются ограничения: рост времени и потребления памяти в браузере при больших объёмах данных, возможные UX-проблемы при длительных вычислениях без перехода в Web Worker, ограничения точности из-за особенностей JS-чисел (floating-point) и необходимость более сложных методов очистки данных, которые сейчас реализованы лишь частично. Централизованная серверная часть решает эти ограничения, но требует инфраструктуры и сетевого соединения. Плюсы гибридного подхода состоят в балансировании интерактивности и вычислительной мощности: пользователь сразу видит результаты простых анализов, а тяжёлые задачи делегируются серверу без «падения» интерфейса.

На основе анализа сформулированы рекомендации по оптимизации: использование CDN для библиотек, кеширование результатов часто повторяющихся запросов, агрегация данных или предварительная фильтрация на клиенте до передачи на сервер, мониторинг клиентской производительности, логирование и трассировка долгих задач на сервере через стандартные инструменты. Обсуждаются перспективы расширения: добавление более сложных моделей ML на сервере, интеграция с базами данных и облачными сервисами для работы с большими данными, мультимодальная визуализация, многопользовательские сценарии совместной работы с сохранением сессий, продвинутые инструменты очистки и подготовки данных.

Таким образом, экспериментальная часть демонстрирует не только работоспособность базовых модулей на реальных данных, но и даёт ясное представление о границах применимости клиентской обработки, сценариях переключения на сервер и путях оптимизации и масштабирования системы, подтверждая практическую ценность разработанного веб-приложения для интерактивного анализа табличных данных и задавая векторы дальнейшего развития.

ЗАКЛЮЧЕНИЕ

Заключение фиксирует, что главная цель работы по разработке и апробации веб-приложения для интерактивного анализа и визуализации табличных данных полностью достигнута: созданная система функционирует непосредственно в браузере, обеспечивая самодостаточный аналитический инструмент с минимальными требованиями к ресурсам и без необходимости установки дополнительного ПО. В рамках поставленных задач был проведён системный анализ существующих решений и архитектур, сравнение клиентских и серверных подходов, обоснован выбор стека Tornado на сервере и JavaScript-библиотек на клиенте, что позволило разработать SPA с тонким бэкендом для отдачи статических ресурсов и опциональными эндпоинтами тяжёлых вычислений при необходимости.

Реализованы ключевые модули: модуль описательной статистики, обеспечивающий расчёт полного набора дескриптивных метрик; модуль инференциальной статистики для проверки гипотез (t-тест, F-тест и непараметрические методы) с гибридным подходом (локальная обработка малых выборок и серверная для больших); модуль машинного обучения (кластеризация k-Means, простая линейная регрессия, kNN) с пороговым переключением на сервер для более ресурсоёмких случаев; модуль визуализации, предоставляющий интерактивные гистограммы, scatter-диаграммы, boxplot и тепловые карты корреляций . Функциональность импорта/экспорта данных реализована через File API, SheetJS, jsPDF, что обеспечивает полный цикл работы: загрузка, анализ, визуализация и получение отчёта.

Валидация на реальных данных (социально-экономические показатели из Росстата) показала высокую точность реализованных алгоритмов: результаты всех аналитических операций совпали с эталонными вычислениями в Python/Pandas/SciPy, что подтверждает корректность математических реализаций на стороне клиента и при делегировании на сервер. Экспериментальные замеры производительности выявили границы интерактивной клиентской обработки: описательная статистика остаётся эффективной до десятков тысяч строк, t-тесты и простая регрессия — до нескольких тысяч записей, k-Means — до трёх–пяти тысяч точек; при превышении этих объёмов приложение автоматически предлагает серверный режим, сохраняя отзывчивость интерфейса. Это подтверждает практическую применимость гибридной архитектуры: интерактивность и автономность при малых/средних данных и мощь сервера для тяжёлых вычислений.

Одновременно выявлены ограничения текущей реализации: клиентская обработка страдает при очень больших наборах из-за ограничений памяти браузера и особенностей JS (floating-point), отсутствуют продвинутые инструменты очистки данных (удаление дубликатов, преобразование типов) и проверки статических допущений (нормальность, гомоскедастичность), что может приводить к некорректным выводам без дополнительной валидации.

Пользовательский опыт при «грязных» данных ограничен текущей обработкой: требуется доработка функционала предобработки и очистки.

На основе опыта сформулированы рекомендации по оптимизации и дальнейшему развитию: внедрение виртуального рендеринга таблиц для больших объёмов, расширение клиентских инструментов очистки данных, проверка статистических предпосылок (например, тест Шапиро–Уилка), добавление более робастных методов анализа, агрегация и предварительная фильтрация данных перед визуализацией, использование Web Workers и WebAssembly для повышения производительности, а главное — создание гибридной модели вычислений, при которой при загрузке крупных файлов или сложных алгоритмов задачи автоматически перенаправляются на серверный эндпоинт с Pandas и scikit-learn, возвращающий готовый результат . Также предлагается интеграция с облачными сервисами и базами данных для работы с большими данными, многопользовательские сценарии с сохранением сессий, мониторинг и логирование серверных метрик, использование CDN для клиентских библиотек и кеширование повторяющихся вычислений.

Практическая значимость заключается в доступности инструмента для широкого круга пользователей (исследователей, аналитиков, студентов), позволяющего проводить полный цикл анализа табличных данных без установки ПО и с минимальными инфраструктурными затратами. Образовательная ценность проявляется в интерактивной демонстрации статистических и ML-методов в привычной среде браузера, способствуя пониманию и вовлечённости. Разработанная архитектура и реализация закладывают основу для масштабирования и интеграции с корпоративными системами и облачными платформами.

Таким образом, заключение фиксирует, что поставленные задачи полностью решены, демонстрируя концептуальную состоятельность веб-приложения как самодостаточной среды для интерактивного анализа данных, и указывает направления дальнейшего развития для преодоления выявленных ограничений и расширения функциональности с учётом гибридной архитектуры.