Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Зав. кафедрой ИТиСУ
Е.В. Кислицын
(подпись) (Ф.И.О.)
«03» 06 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

АВТОМАТИЗАЦИЯ РУЧНОГО ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЙ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Научный руководитель: Коломыцева Анна Олеговна
к.э.н., доцент
подпись

Нормоконтролер: Бредихина Наталья Сергеевна
подпись

Консультант: Павлов Марк Владимирович
подпись

Студент группы: РИМ-220908 Никитин Александр Алексеевич
подпись

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления
Направление подготовки 09.04.01 Информатика и вычислительная техника
Образовательная программа Инженерия искусственного интеллекта

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Никитин Александр Алексеевич группы РИМ-220908
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы Автоматизация ручного тестирования web-приложений на основе искусственного интеллекта
Утверждена распоряжением по институту от «4» декабря 2023 г. № 33.02-05/298
2. Научный руководитель Коломыцева Анна Олеговна, доцент, кандидат экономических наук
(Ф.И.О., должность, ученая степень, ученое звание)
3. Исходные данные к работе полученные в ходе преддипломной практики
4. Перечень демонстрационных материалов презентация в MS PowerPoint
5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	Глава 1. Теоретические основы тестирования веб-приложений	до 23.03.2024 г.	
2.	Глава 2. Разработка и выбор алгоритма машинного обучения для автоматизации тестирования веб-приложений	до 29.04.2024 г.	
3.	Глава 3. Разработка решения и оценка продуктовой эффективности	до 19.05.2024 г.	
4.	ВКР в целом	до 20.05.2024 г.	

6. Консультанты по работе с указанием относящихся к ним разделов*

Раздел	Консультант	Подпись, дата	
		задание выдал	задание принял
Раздел 2	Павлов М.В.	10.05.2024	10.05.2024

Научный руководитель Коломыцева А.О. _____
Ф.И.О. (подпись)
Студент задание принял к исполнению 4.12.23 _____
дата (подпись)
7. Допустить Никитина Александра Алексеевича к защите выпускной квалификационной работы в экзаменационной комиссии
Зав. кафедрой ИТиСУ _____ Е.В. Кислицын
(подпись) Ф.И.О.

РЕФЕРАТ

Магистерская диссертация 79 с., 14 рис., 6 табл., 56 источн., 1 прил. АВТОМАТИЗАЦИИЯ ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ, ЦЕПИ МАРКОВА, LSTM, ГЕНЕРАЦИЯ ТЕСТ КЕЙСОВ, РАЗРАБОТКА СИСТЕМЫ.

Объектом данного исследования является процессы тестирования web-приложений.

Предметом исследования является: автоматизация процессов тестирования web приложений за счет внедрения искусственного интеллекта.

Цель работы – разработать систему автоматизации тестирования для web-приложений, за счет которой будет повышена скорость выполнения задач и принятие управленческих решений, связанных с качеством конечного продукта.

Методы исследования – литературный обзор, анализ современных методов автоматизации тестирования, машинное обучение.

Результатом работы является система для автоматической генерации тестовых сценариев.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тестирование является неотъемлемой частью современной разработки любых решений в области информационных технологий и не только. Ведь тестирование является показателем качества выпускаемого продукта. От этого компонента современной разработки зависит на сколько “чистый” продукт получат пользователи каких-либо систем и приложений, будь то это web-проект, мобильное приложение или десктоп программой.

Современное тестирование направлено на обнаружение и устранение как можно большего числа ошибок и по возможности, чем раньше будет обнаружена ошибка, тем “дешевле” для разработки продукта будет она исправлена. Следствием такой деятельности является повышение качества ПО по всем его характеристикам. Существующие на сегодняшний день методы тестирования программного обеспечения не позволяют однозначно и полностью устранить все дефекты и ошибки и установить корректность функционирования программного продукта. Поэтому, все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного продукта.

Разработка ПО сложный и многогранный процесс, в ходе которого тысячи строк кода объединяются в единое целое. И главное тут обеспечить надежность, стабильность и безопасность программы перед ее выпуском на рынок. Недостаточное тестирование может привести к выходу продукта с критическими ошибками. Стоит отметить, что тестирование — это очень гибкий и даже творческий процесс, который требует использования различных методологий в зависимости от направления, тестируемого продукт. Общая проблематика автоматизации тестирования специализированного программного обеспечения является возможная неэффективность подходов, основанных на применении генетических алгоритмов, сильно зависит от выбранного критерия полноты. Необходим анализ эффективности некоторых

известных критериев полноты тестовых наборов при применении генетических алгоритмов для генерации тестов. А также верификация представляет собой процесс анализа программного объекта для обнаружения различий между существующим и необходимым состоянием (то есть, ошибки) и оценивания особенностей программного объекта. Тестирование программного обеспечения является деятельностью, которая должна производиться в течение всего процесса разработки.

Актуальность работы обусловлена необходимостью усовершенствования процессов автоматизации тестирования программного обеспечения web-приложений с помощью применения искусственного интеллекта. Это позволит ускорить процесс тестирования и тем самым сократить стоимость разработки и увеличить качество выпускаемого продукта.

Объект: процессы тестирования web-приложений

Предмет: автоматизация процессов тестирования web приложений за счет внедрения искусственного интеллекта.

Цель: разработать систему автоматизированного тестирования для web-приложений, за счет которой будет повышена скорость выполнения задач и принятие управленческих решений, связанных с качеством конечного продукта.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– Провести анализ существующих подходов к тестированию ПО;

– описать специфику автоматизации тестирования ПО для web-приложений;

– провести теоретическое исследование ML-алгоритмов, применяемыхдля решений задач в тестировании для web-приложений;

– провести практический анализ ML-алгоритмов;

– описать и обосновать выбранный алгоритм ML для реализации

разработанной модели;

– провести анализ построения выбранной модели для выполнения задачи с учетом всех особенностей;

– подготовить данные для обучения модели и провести анализ;

– выполнить процесс разработки и тестирование предлагаемогорешения;

– провести экономическую оценку продуктовой эффективности модели, основанной на автоматизации тестирования с помощью ИИ.

Научная новизна работы заключается в том, что усовершенствованы процессы автоматизированного тестирования для web-приложениях.

1 Теоретические основы тестирования веб-приложений

1.1 Анализ существующих подходов к тестированию ПО

Для начала стоит обозначить что же такое тестирование. Это процесс анализа ПО, направленный на выявление отличий между его реально существующими и требуемыми свойствами (дефект) и на оценку свойств ПО. Так же в жизненном цикле ПО определены среди прочих вспомогательные процессы верификации, аттестации, совместного анализа и аудита. Процесс верификации является процессом определения того, что программные продукты функционируют в полном соответствии с требованиями или условиями, поставленными в техническом задании. Данный процесс может включать анализ, проверку и тестирование. Процесс аттестации является процессом определения полноты соответствия установленных требований, созданной системы или программного продукта их функциональному назначению. Процесс совместного анализа является процессом оценки состояний и, при необходимости, результатов работ по проекту. Тестирование основывается на тестовых процедурах с конкретными входными данными, начальными условиями и ожидаемым результатом, разработанными для определенной цели, такой, как проверка отдельной программы или верификация соответствия на определенное требование. Тестовые процедуры могут проверять различные аспекты функционирования программы — от правильной работы отдельной функции до адекватного выполнения бизнес-требований. Какие инструментальные средства будут использоваться для поиска и для документирования найденных дефектов. Стоит всегда помнить о тестировании, так как это неотъемлемая часть современной разработки, чем раньше дефект будет найдет, тем быстрее, а значит дешевле он будет исправлен.

Существует множество современных подходов к тестированию

программного обеспечения независимо от его предназначения. Основа тестирования, как процесса является нахождение дефектов, локализация и их исправление на стадии разработки системы до выпуска ее конечному пользователю [32].

Для того, чтобы описать существующие подходы к тестированию программного обеспечения необходимо понимать этапы и последовательность современной разработки продукта и его методологию.

Современные средства разработки программ много обеспечения и технологии позволяют создание и развитие чрезвычайно сложных, распределенных программных систем, взаимодействующих с различными внешними агентами. С использованием новых средств и методов объектно-ориентированного программирования появилась возможность создания более сложных программ. Благодаря развитию возникли новые задачи более высокого уровня абстракции, требующие своего решения. К наи более важным задачам относятся:

– обеспечение эффективной распределенной работы множества объектов;

– поддержка транзакций;

– безопасность объектно-ориентированной системы;

– долговечное сохранение данных;

– разрешение нелокальных эффектов поведения объектов в контексте сложной распределённой системы.

Этапы жизненного цикла программного обеспечения существуют на примере алгоритма Software Life Cycle Model (SLCM) [14]. Основным и наиболее популярным подходом к разработке ПО является именно SLCM. SLCM – это алгоритм создания IT-продукта, который состоит из 6 этапов и охватывает период с момента принятия решения о его разработке и заканчивается, когда ПО перестают использовать. Каждый этап опирается на результат предыдущего и дает пул необходимых указаний для выполнения

последующего. Основная функция SLCM – регламентирование и формализация процесса разработки [40]. Это важно при командной работе, когда задействуют десятки специалистов.

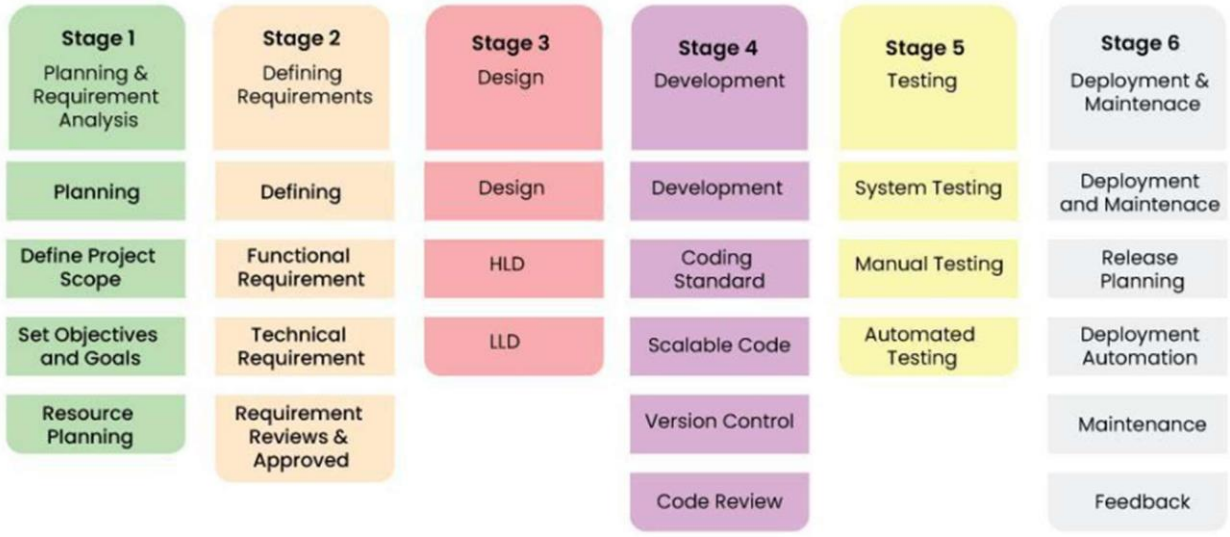


Рисунок 1 – Алгоритм Software Life Cycle Model

Исходя из рисунка видно, что по алгоритму SLCM, который показывает жизненный цикл разработки программного обеспечения выделено 6 этапов. Стоит рассмотреть каждый из них для понимания, как создается конечный продукт. Первым этапом является – планирование и анализ требований у заказчика. На этом этапе разрабатывается базовый проект, исходя из всей доступной информации на данный момент.

Следующий этап – это определение требований, здесь аналитик и команда разработки выбирают наиболее подходящие решения для реализации проекта, так же идет написание документации для дальнейшей разработки и тестирования системы.

Далее идет проектирование архитектуры - здесь исходя из требований, которые были сформированы на предыдущем этапе, разрабатывается архитектура программного обеспечения.

Следующий этап – это непосредственно разработка продукта. Команда

программистов пишет код и отлаживает методы, которые были заложены в функциональных требованиях и в техническом задании.

После разработки идет непосредственно этап тестирования иинтеграции продукта, здесь идет поиск, локализация и исправление дефектов.Проведение ручных и автоматизированных тестов для проверки работы системы.

Конечный этап разработки является – развертывание и поддержка продукта. После того, как тестировщик разрешает команде выпускать продукт, то идет процесс интеграции продукта или новой функциональности общий доступ для всех пользователей [41].

Разработка программного обеспечения на основе жизненного цикла разработки программного обеспечения является важной основой для более качественной и структурированной разработки ПО.

Другими словами все, что было описано выше можно назвать итерациями. Итерация — это законченный цикл разработки, приводящий к выпуску конечного продукта или некоторой его сокращенной версии, которая расширяется от итерации к итерации, чтобы, в конце концов, стать законченной системой. Каждая итерация включает, как правило, задачи планирования работ, анализа, проектирования, реализации, тестирования и оценки достигнутых результатов. Однако соотношения этих задач может существенно меняться. В соответствие с соотношением различных задач в итерации они группируются в фазы. В первой фазе, начало — основное внимание уделяется задачам анализа. В итерациях второй фазы — разработка — основное внимание уделяется проектированию и опробованию ключевых проектных решений. В третьей фазе — построение — наиболее велика доля задач разработки и тестирования. А в последней фазе — передача — решаются в наибольшей мере задачи тестирования и передачи системы заказчику. Каждая фаза имеет свои специфические цели в жизненном цикле продукта и считается выполненной, когда эти цели достигнуты. Все итерации, кроме, итераций фазы начало, завершаются созданием функционирующей версии

разрабатываемой системы [25].

Стоит отметить какое же место занимает тестирование в процессах разработки ПО и какие задачи выполняет.

В задачи тестирования входят:

– комплексный контроль качества;

– подготовка тестовой документации;

– обнаружение и локализация ошибок в функционировании программных продуктов;

– фиксирование и отслеживание ошибок в функционировании программных средств;

– проверка соответствия документации программного продукта стандартам и реально реализованным функциям;

– участие в разработке и внедрении системы качества;

– оценка производительности разрабатываемых программных средств на различных программно-аппаратных платформах и их специфических конфигурациях.

Внутри производственной команды технологические процессы, как правило, регламентированы внутренними документами или внутрикорпоративными стандартами. Они представляют собой технологический цикл производства программного средства. Типичных технологических цепочек внутри компании — разработчика программного обеспечения большое множество [35].

После того, как была показана основная методология разработки ПО стоит перейти непосредственно к анализу и описанию существующих методологий тестирования программного обеспечения. В данный момент существуют различные подходы к тестированию все зависит от процессов разработки и тестируемых систем в целом. Также все зависит с чем связан проект и на какой платформе развернута система – это может быть мобильное

приложение, веб-приложение или же стационарная (Desktop) программа. В зависимости от условий тестирования выбирается и методология, но рассмотрим наиболее популярные и основные. Для начала стоит вообще

понимать для чего нужно тестирование. Цель тестирования — это нахождение багов до того, как их найдут пользователи. Еще одной немаловажной целью тестирования заключается в том, что тестирование призвано улучшить ПО [14]. А теперь исследуем методы тестирования, используемые для выявления ошибок. Методы тестирования можно разделить на группы

По объекту тестирования:

– Функциональное тестирование – направлено на проверку корректностиработы функциональности приложения.

– Избыточное тестирование – тестирование приложения со всеми условиями выполнения. Может применяться для проверки простых компонентов.

– Тестирование производительности – исследует показатели скорости реакции приложения на внешние воздействия при нагрузке.

– Тестирование данных и баз данных – направлено на исследование характеристик данных, например, полнота, непротиворечивость, цельность, структурированность.

– Тестирование удобства использования – настроено на исследование того, насколько конечному пользователю понятно, как работать с продуктом, а также на то, насколько ему приятно использовать продукт.

– Тестирование интерфейса пользователя – направлено на проверку интерфейсов приложения или его компонентов. (Обнаруживает ошибки интерфейса).

– Тестирование безопасности – проверяет способности приложения противостоять хакерским попыткам получения доступа к данным или

функциям.

– Тестирование локализации – исследует готовность продукта к работе с использованием различных языков и с учетом различных национальных и культурных особенностей.

– Тестирование совместимости – направлено на проверку способности приложения, работать в указанном окружении.

По знанию системы:

– Тестирование чёрного ящика – доступа к коду нет, тестировщик оказывает на продукт воздействия и проверяет реакцию тем же способом, каким при реальной эксплуатации приложения на него воздействовали бы пользователи. В рамках тестирования по методу чёрного ящика основной информацией для создания тест-кейсов выступает документация и общий здравый смысл.

– Тестирование белого ящика – доступ к коду есть, разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого ПО. При тестировании белого ящика используются метрики покрытия кода.

– Тестирование серого ящика – имеется доступ к части кода.

По степени автоматизации:

– Ручное тестирование – тест-кейсы выполняет человек.

– Автоматизированное тестирование – тест-кейсы частично или полностью выполняет специальное инструментальное средство, что позволяет исключить человека из выполнения некоторых задач в процессе тестирования.

По уровню тестирования:

– Компонентное (модульное) тестирование – проверяются отдельные небольшие части приложения, например, отдельный класс, функция или небольшие библиотеки.

– Интеграционное тестирование – проверяется взаимодействие между

несколькими частями приложения, каждая из которых проверена отдельно на стадии модульного тестирования. При наличии резервавремени на данной стадии тестирование ведётся итерационно, спостепенным подключением последующих подсистем.

– Регрессионное тестирование – приложение проверяется как единое целое. Выявляются дефекты "на стыках" компонентов, а также есть возможность полноценного взаимодействия с программой с точки зрения конечного пользователя [37].

Исходя из всего вышеперечисленного, можно сделать вывод, что существует множество подходов к тестированию ПО, все зависит от непосредственно системы, а уже после выбирается подходящая методология.

1.2 Описание специфики автоматизации тестирования ПО для web-приложений

Автоматизация тестирования web-приложений — это процесс использования специализированных инструментов и скриптов для автоматического выполнения тестовых сценариев и проверки функциональности, производительности и безопасности web-приложений. Стоит выделить несколько ключевых аспектов специфики автоматизации тестирования программного обеспечения web-приложений [9]:

- веб-ориентированный подход;
- многоуровневость тестирования;
- кросс браузерное и кросс платформенное тестирование;
- тестирование производительности;
- безопасность;
- интеграция с CI/CD

Для понимания, как работает автоматизация тестирования стоит

подробно их описать и в дальнейшем показать, как работает автоматизация.

Веб-ориентированный подход. Данный подход к автоматизации тестирования программного обеспечения — это методология, разработанная специально для учета особенностей веб-приложений и их компонентов. Этот подход является ответом на уникальные технологические и функциональные характеристики web-приложений, а также на требования и ожидания пользователей к их производительности, надежности и безопасности. Основные принципы веб-ориентированного подхода включают в себя: интерактивность и пользовательский опыт. Web-приложения являются высоко интерактивными, что предполагает активное взаимодействие пользователей с интерфейсом приложения через браузеры. Чтобы обеспечить качественный пользовательский опыт, автоматизация тестирования должна быть способна точно воспроизводить действия пользователя, такие как ввод данных, нажатие кнопок, навигация по странице и взаимодействие с элементами интерфейса со стороны пользователя. Немаловажна работа с элементами страниц. Web-приложения построены на HTML, CSS и JavaScript, что делает их структуру динамичной и изменчивой. Поэтому автоматизация тестирования должна уметь обнаруживать и взаимодействовать с различными элементами веб-страниц, такими как кнопки, поля ввода, выпадающие списки и т. д., и учитывать их изменения и обновления.

Таким образом, веб-ориентированный подход к автоматизации тестирования программного обеспечения для web-приложений — это комплексный и универсальный подход, позволяющий обеспечить высокое качество и надежность web-приложений в условиях быстро развивающихся и меняющихся требований пользователей.

Следующим подходом к автоматизации тестирования является многоуровневость тестирования. Многоуровневое тестирование программного обеспечения — это методология, которая охватывает несколько уровней архитектуры приложения и проверяет его функциональность,

производительность и безопасность на каждом из этих уровней. В контексте веб-приложений это включает в себя клиентскую часть (frontend), серверную часть (backend) и базу данных. На клиентском уровне тесты направлены на проверку пользовательского интерфейса, его взаимодействия с пользователем и корректности отображения данных. Средства автоматизации тестирования интерфейса, такие как Selenium WebDriver и Cypress, используются для воспроизведения действий пользователя и проверки реакции интерфейса на различные события. Серверная сторона тестируется на соответствие бизнес-логике, обработку запросов и ответов, а также взаимодействие с базой данных. Тестирование API и обработки ошибок часто выполняется с помощью таких инструментов, как Postman и REST API. Web-приложения являются высоко интерактивными, что предполагает активное взаимодействие пользователей с интерфейсом приложения через браузеры. Чтобы обеспечить качественный пользовательский опыт, автоматизация тестирования должна быть способна точно воспроизводить действия пользователя, такие как ввод данных, нажатие кнопок, навигация по странице и взаимодействие с элементами интерфейса со стороны пользователя. Немаловажна работа с элементами страниц. Интеграционные тесты направлены на проверку взаимодействия между компонентами приложения, включая взаимодействие клиент-сервер и взаимодействие с внешними сервисами или системами. Здесь используются такие инструменты интеграционного тестирования, как Apache JMeter и WireMock. Этот подход к тестированию обеспечивает всесторонний охват различных аспектов функциональности, производительности и безопасности веб-приложений, помогая повысить их качество и надежность в условиях быстрого развития и меняющихся требований пользователей [5].

Этот подход к тестированию обеспечивает всесторонний охват различных аспектов функциональности, производительности и безопасности веб-приложений, помогая повысить их качество и надежность в условиях быстрого развития и меняющихся требований пользователей.

Также немаловажным элементом тестирования является тестирование безопасности, которое тоже можно автоматизировать. Специфика автоматизации тестирования безопасности программного обеспечения включает в себя ряд функций, направленных на выявление и устранение уязвимостей и защиту от потенциальных атак. При использовании автоматизированных средств тестирования безопасности для выявления уязвимостей проводится сканирование кода, анализ уязвимостей и тестирование структуры web-приложения. Важным аспектом является тестирование механизмов аутентификации и авторизации для проверки корректности доступа пользователей к ресурсам приложения. Также проверяется защита от атак на переполнение буфера и других типов атак, которые могут привести к нарушению работы приложения или утечке конфиденциальных данных. При взаимодействии с другими системами через API или веб-сервисы проверяется безопасность этого взаимодействия, включая корректность авторизации и аутентификации, а также обработки входных данных. Интеграция тестов безопасности в процессы непрерывной интеграции и доставки гарантирует, что эти тесты будут выполняться регулярно и автоматически после каждого изменения кода и развертывания приложения. Такой подход позволяет обнаруживать и устранять уязвимости на ранних этапах разработки и обеспечивать высокий уровень защиты приложения от потенциальных угроз.

Немаловажным моментом для понимания что же такое автоматизированное тестирование, важно понимать его преимущества и недостатки. Для того, чтобы избежать неэффективного применения автоматизации, следует обходить ее недостатки и максимально использовать преимущества. Преимущества автоматизации тестирования:

– повторяемость – все написанные тесты всегда будут выполняться однообразно, то есть исключен «человеческий фактор». Тестировщик не пропустит тест по неосторожности и ничего не напутает в результатах;

– быстрое выполнение – автоматизированному скрипту не нужно сверяться с инструкциями и документациями, это сильно экономит время выполнения; меньшие затраты на поддержку – когда автоматические скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время чем на проведение того же объема тестирования вручную;

– отчеты – автоматически рассылаемые и сохраняемые отчеты о результатах тестирования;

– выполнение без вмешательства – во время выполнения тестов инженер-тестировщик может заниматься другими полезными делами, или тесты могут выполняться в нерабочее время (этот метод предпочтительнее, так как нагрузка на локальные сети ночью снижена).

Недостатки автоматизации тестирования (их тоже немало):

– повторяемость – все написанные тесты всегда будут выполняться однообразно. Это одновременно является и недостатком, так как тестировщик, выполняя тест вручную, может обратить внимание на некоторые детали и, проведя несколько дополнительных операций, найти дефект. Скрипт этого сделать не может;

– затраты на поддержку – несмотря на то, что в случае автоматизированных тестов они меньше, чем затраты на ручное тестирование того же функционала – они все же есть. Чем чаще изменяется приложение, тем они выше;

– большие затраты на разработку – разработка автоматизированных тестов — это сложный процесс, так как фактически идет разработка приложения, которое тестирует другое приложение. В сложных автоматизированных тестах также есть фреймворки, утилиты, библиотеки и прочее. Естественно, все это нужно тестировать и отлаживать, а это требует времени [2].

Исходя из всех перечисленных преимуществ и недостатков стоит

понимать, что нет идеальной системы, которая могла бы разработать, протестировать и выпустить пользователем идеальное приложение, но благодаря автоматизации, можно сократить и улучшить выпускаемый продукт.

1.3 Теоретическое исследование ML-алгоритмов в применении к тестированию web-приложений

Алгоритмы ML (машинного обучения) достаточно уверенно входят в повседневную жизнь не только обычного человека, но и в целые системы разработок и планирования в крупнейших компаниях мира. Сейчас практически любая большая компания применяет искусственный интеллект и машинное обучение для автоматизации своих рутинных процессов – это позволяет сократить время и деньги на какие-то вещи, которые раньше мог делать только человек. Например, сбор и составление квартальных отчетов, написание скриптов, тестирование некоторых участков системы. Для автоматизации тестирования web-приложений можно выделить несколько основных подходов и технологий, которые позволят увеличить тестируемое покрытие продукта и улучшить его качество. Рассмотрим их и проведеманализ для выбора наиболее подходящей технологии.

Использование инструментов для записи и воспроизведения действий пользователя. В современном мире автоматизация тестирования веб-приложений становится все более важной и актуальной задачей. Одним из методов автоматизации, который привлекает внимание и находит широкое применение, является использование инструментов для записи и воспроизведения действий пользователя. Этот метод является эффективным инструментом для создания автоматизированных тестовых сценариев,

основанных на реальных действиях пользователя. Эта технология позволяет записывать действия пользователя при взаимодействии с web-

приложением иавтоматически воспроизводить их во время тестирования. С применением технологий машинного обучения такие инструменты могут быть усовершенствованы для распознавания изменений в пользовательском интерфейсе и автоматической корректировки тестовых сценариев при обновлении приложения, так же при исправлении дефектов разработчиками такая технология будет очень полезной для воспроизведения багов после тестирования в автоматическом режиме для ускорения понимания сути ошибки и соответственно ее исправления.

Принцип работы данного инструмента записи и воспроизведения действий пользователя заключается в том, что в процессе тестирования записываются все действия, которые пользователь совершает при взаимодействии с web-приложением. Это может быть нажатие на кнопки, взаимодействия с пагинацией страниц, заполнение форм, переход по ссылкам и другие действия. Полученные данные сохраняются в специальном формате, который затем может быть использован для воспроизведения этих действий в автоматическом режиме для таксировщиков и разработчиков в команде [6].

Одним из главных преимуществ использования такого инструмента для записи и воспроизведения действий пользователей является повышение скорости и эффективности процесса тестирования, а именного локализация ошибок, которые впоследствии будут исправлены разработчиками. Записанные тестовые сценарии могут быть легко воспроизведены несколько раз без необходимости повторного ручного ввода данных. Это значительно сокращает время, затрачиваемое на тестирование, и повышает его надежность. Кроме того, использование подобных инструментов позволяет создавать более стабильные и надежные тестовые сценарии. Так как действия пользователя записываются точно в момент их выполнения, тестовыесценарии получаются более стабильными и надежными.

Отсюда следует ключевые функции и положительные эффекты этого алгоритма:

– запись действий пользователя;

– воспроизведение действий пользователя;

– проверка реакции приложения;

– распознавание изменений;

– повышение точности воспроизведения;

– интеграция с другими инструментами.

Небольшой вывод по применению этого алгоритма, таким образом, использование инструментов для записи и воспроизведения действий пользователя является эффективным и перспективным подходом к автоматизации тестирования web-приложений. Он позволяет ускорить процесс тестирования, повысить его надежность и обеспечить высокое качество разрабатываемого программного обеспечения.

– Использование компьютерного зрения для обнаружения элементов интерфейса.

Одним из таких методов, который привлек внимание и активно применяется в отрасли, является использование компьютерного зрения для обнаружения элементов интерфейса web-приложений. Суть этого подхода заключается в том, что система компьютерного зрения обучается распознавать элементы интерфейса веб-приложения на основе набора обучающих данных. Эти данные обычно содержат изображения различных элементов интерфейса, таких как кнопки, текстовые поля, флажки и другие [15]. С помощью различных алгоритмов машинного обучения, таких как сверточные нейронные сети (CNN), модель обучается извлекать признаки из изображений и классифицировать их. После обучения модели разрабатывается система для обнаружения элементов интерфейса в веб-приложении. Эта система включает в себя алгоритмы обработки изображений, извлечения признаков и классификации. Во время автоматизированного тестирования система компьютерного зрения сканирует визуальное представление веб-приложения

и обнаруживает элементы интерфейса на экране. Обнаруженные элементы затем сопоставляются с ожидаемыми элементами, определенными в сценариях тестирования. Одним из главных преимуществ использования компьютерного зрения в тестировании веб-приложений это - возможность автоматизировать процесс тестирования визуального представления приложения. Это позволяет значительно сократить время, затрачиваемое на тестирование, и повысить его эффективность. Кроме того, система компьютерного зрения может быть использована для обнаружения различных дефектов интерфейса, таких как неправильное расположение элементов в верстке сайта или нарушение цветовой схемы.

Отсюда следует ключевые функции и положительные эффекты этого алгоритма:

- обучение модели распознавания элементов интерфейса;
- разработка системы распознавания элементов интерфейса;
- применение в автоматизированных тестах;
- визуальная проверка согласованности;
- обнаружение дефектов в визуальном интерфейсе.

Таким образом, использование компьютерного зрения для обнаружения элементов интерфейса при тестировании веб-приложений является перспективным и эффективным подходом, способным повысить качество и надежность разрабатываемого программного обеспечения. Применение алгоритмов машинного обучения для генерации тестовых данных.

По мере развития web-приложений и увеличения их сложности возникает необходимость в эффективных методах генерации тестовых данных для обеспечения их качества и надежности. Одним из подходов, который привлекает внимание и находит широкое применение, является использование алгоритмов машинного обучения для генерации тестовых данных и выявить потенциальные проблемы. Суть этого подхода заключается в использовании моделей машинного обучения, обученных на существующих данных, для

генерации новых тестовых данных. Обученные модели способны анализировать характеристики приложения и генерировать различные сценарии использования, учитывая различные аспекты его функциональности и поведения. Одним из ключевых преимуществ использования алгоритмов машинного обучения для генерации тестовых данных является возможность создания разнообразных и репрезентативных наборов данных, охватывающих различные сценарии использования веб-приложения. Это позволяет улучшить тестовое покрытие тем самым помогает выявить максимальное количество возможных дефектов на более раннем этапе разработки. Кроме того, использование алгоритмов машинного обучения позволяет автоматизировать процесс генерации тестовых данных, что повышает его эффективность и результативность. Автоматизация процесса также обеспечивает актуальность и разнообразие генерируемых данных, что является важным аспектом при тестировании динамичных иизменяющихся web-приложений.

Для оценки качества сгенерированных данных можно использовать различные метрики, такие как охват функциональности приложения, разнообразие сценариев использования, соответствие ожидаемым результатам и другие. Это позволяет убедиться в том, что тестовые данныерепрезентативны и пригодны для использования в процессе тестирования.

Отсюда следует ключевые функции и положительные эффекты этого алгоритма:

- модели обучения на основе имеющихся данных;
- генерация новых тестовых данных;
- автоматизация процесса генерации тестовых данных;
- использование показателей качества данных.

Таким образом, использование алгоритмов машинного обучения для генерации тестовых данных является эффективным и перспективным подходом к повышению качества и надежности тестирования web-

приложений, например, распознавание и анализ текста.

Одним из эффективных методов обеспечения качества текста является использование алгоритмов распознавания и анализа текста, основанных на технологиях обработки естественного языка (NLP). Модели машинного обучения для обработки естественного языка могут использоваться для анализа текстовых сообщений, введенных пользователями в приложение (например, комментариев, отзывов), что позволяет автоматически выявлять потенциальные проблемы или несоответствия. Это включает в себя автоматическую проверку правильности орфографии и грамматики, поиск и анализ ключевых слов и фраз, а также анализ текстовых сообщений, предупреждений и инструкций на предмет соответствия ожидаемым стандартам и требованиям. Алгоритмы распознавания и анализа текста также позволяют автоматически анализировать комментарии и отзывы пользователей о web-приложении. Это помогает выявить потенциальные проблемы или недостатки приложения, обнаруженные пользователями, и предпринять соответствующие действия для их устранения [18].

Кроме того, использование алгоритмов NLP при тестировании web-приложений позволяет автоматизировать процесс генерации тестовых данных. Это включает в себя создание различных комбинаций текстовых входных данных для тестирования различных сценариев использования приложения, что позволяет увеличить охват тестированием и выявить потенциальные проблемы, связанные с обработкой и анализом текстовых данных.

Отсюда следует ключевые функции и положительные эффекты этого алгоритма:

- автоматизация тестирования текстового содержимого
- анализ ключевых слов и фраз;
- проверка правильности орфографии и грамматики;
- анализ пользовательских комментариев и обратной связи;

– автоматизированная генерация тестовых данных.

Таким образом, использование алгоритмов распознавания и анализа текста при тестировании web-приложений является эффективным способом повышения качества и надежности приложения, а также обеспечения соответствия потребностям и ожиданиям пользователей. Прогнозирование критических областей.

Прогнозирование критических областей - важный аспект тестирования web-приложений, направленный на выявление и устранение потенциальных проблем или уязвимостей, которые могут повлиять на производительность и безопасность приложения. Этот подход основан на анализе структуры, кода и архитектуры веб-приложения с целью выявления уязвимостей, в которых наиболее вероятно возникновение ошибок или неполадок. Одним из ключевых инструментов, используемых для прогнозирования критических областей, являются алгоритмы машинного обучения. Эти алгоритмы могут анализировать различные параметры и метрики, относящиеся к компонентам web-приложений, для определения их потенциальной критичности. Например, они могут оценить сложность кода, уровень зависимостей между компонентами, степень обновлений и изменений в той или иной области, а также множество других факторов [40].

Алгоритмы машинного обучения могут использоваться для прогнозирования областей приложения, которые наиболее вероятно будут подвержены сбоям или ошибкам при изменениях кода или добавлении новых функций. Это позволяет усилить усилия по тестированию в этих областях. Анализируя данные и применяя алгоритмы машинного обучения, можно оценить вероятность возникновения проблем или ошибок в определенных областях web-приложения. Это позволяет определить критические области, где риск возникновения проблем наиболее высок, и сосредоточиться на их тестировании и улучшении. Прогнозирование критических областей также позволяет определить приоритеты тестирования. Это помогает сосредоточить

ресурсы и усилия на тестировании и улучшении наиболее важных компонентов приложения, что способствует повышению его надежности и стабильности. Прогнозирование критических областей также включает мониторинг и анализ результатов после тестирования. Это позволяет оценить эффективность прогнозирования критических областей, выявить новые проблемы или уязвимости и определить необходимость дальнейших улучшений и корректировок.

Отсюда следует ключевые функции и положительные эффекты этого алгоритма:

- анализ и выявление уязвимых точек;
- использование алгоритмов машинного обучения;
- оценка вероятности возникновения проблем;
- приоритизация тестирования;
- мониторинг и анализ результатов.

Таким образом, прогнозирование критических областей при тестировании веб-приложений является эффективным подходом, позволяющим более точно выявлять и устранять потенциальные проблемы и уязвимости, повышая надежность и качество работы приложения.

Анализ алгоритмов ML и их применение в тестирования web-приложений. В этом пункте были рассмотрены различные алгоритмы машинного обучения, такие как классификация, регрессия, кластеризация и другие, и их применение для автоматизации тестирования веб-приложений [44]. Алгоритмы машинного обучения могут использоваться для прогнозирования областей приложения, которые наиболее вероятно будут подвержены сбоям или ошибкам при изменениях кода или добавлении новых функций. Это позволяет усилить усилия по тестированию в этих областях. Анализируя данные и применяя алгоритмы машинного обучения, можно оценить вероятность возникновения проблем или ошибок в определенных областях web-приложения. Применение алгоритмов машинного обучение в

тестировании, а также одним из ключевых аспектов этого преимущества является возможность автоматической адаптации алгоритмов к изменениям в приложении, таким как добавление новой функциональности, изменение интерфейса или оптимизация кода. В таблице 1 перечислены ключевые алгоритмы для тестирования с помощью машинного обучения. С помощью из можно применять автоматизацию тестирования и улучшать выпускаемый продукт.

Таблица 1 - Ключевые алгоритмы для тестирования с помощью ML

Алгоритмы	Применение в тестировании web-приложений
Автоматизация анализа текстового содержимого (NLP)	Автоматизация текстовогосодержимого Поиск и анализ ключевых слов и фраз Проверка орфографии и грамматики Анализ пользовательских комментариев
Алгоритмы компьютерного зрения	Распознавание и анализ интерфейса
Алгоритмы машинного обучения	Генерация тестовых данных

В таблице, которая представлена выше представлены ключевые алгоритмы, которые могут быть применены для тестирования web-приложений с помощью машинного обучения.

Теоретическая база показала, что использование ML-моделей позволяет повысить эффективность процесса тестирования, повысить качество обнаружения и исправления дефектов, а также сократить время, затрачиваемое на разработку и выполнение тестовых сценариев. Более того, алгоритмы ML демонстрируют способность адаптироваться к изменяющейся среде web-приложений и обнаруживать новые типы проблем, что делает их

важным инструментом для обеспечения надежности и устойчивости web-приложений.

Однако, несмотря на значительные достижения в этой области, остается ряд вызовов и проблем, которые требуют дальнейших исследований. К ним относятся повышение точности и надежности ML-моделей, адаптация алгоритмов к специфике различных типов веб-приложений, а также разработка эффективных методов оценки качества и результативности ML-технологий при тестировании. В целом, результаты исследования указывают на перспективность и важность использования алгоритмов ML при тестировании web-приложений и обеспечивают основу для дальнейших исследований и разработок в этой области [51]. Эти подходы могут быть использованы как самостоятельно, так и в комбинации друг с другом для создания более эффективной системы автоматизированного тестирования web-приложений на основе искусственного интеллекта.

Таким образом, результаты данного исследования подтверждают перспективность применения ML-алгоритмов в тестировании web-приложений и предоставляют основу для дальнейших исследований в этой области.

Немаловажным вопросом является оценка преимуществ и недостатков применения алгоритмов машинного обучения. Далеко не все алгоритмы являются эффективными, но для конкретных ситуация и приложений, как и большинство тестовых сценариев необходимо понимать, что тестируем и какой ожидаемый результат является правильным. Для того, чтобы более подробно рассмотреть работу алгоритмов необходимо понимать их преимущества и недостатки в целом, как применения их для тестирования.

Начнем с преимуществ использования ML в тестировании web-приложений. Самым основным является – автоматизация. Использование алгоритмов машинного обучения позволяет автоматизировать многие аспекты тестирования web-приложений, что сокращает время и ресурсы,

затрачиваемые на этот процесс. Автоматизированное тестирование с использованием алгоритмов машинного обучения предоставляет разработчикам web-приложений ряд преимуществ. Прежде всего, это экономит время и ресурсы. Значительную часть тестирования можно автоматизировать, что сокращает время, затрачиваемое на тестирование каждого обновления web-приложения [54]. Кроме того, автоматизированные тесты, основанные на алгоритмах машинного обучения, обеспечивают более высокую скорость тестирования и повышают его качество за счет более полного охвата различных сценариев использования приложений. Это также способствует более быстрому реагированию на изменения и новые требования к приложениям, что особенно важно в динамичной среде разработки. Таким образом, использование алгоритмов машинного обучения в автоматизации тестирования позволяет разработчикам повысить эффективность, скорость и качество процесса разработки web-приложений.

Еще одним важным преимуществом алгоритмов машинного обучения является способность адаптироваться к изменяющимся условиям и среде web-приложений. Они могут быстро и гибко реагировать на изменения в приложении, включая добавление новой функциональности или изменение интерфейса. Это позволяет поддерживать высокую точность и эффективность тестирования даже в условиях быстрой и динамичной разработки. Кроме того, использование алгоритмов машинного обучения позволяет оптимизировать процесс тестирования, устраняя неэффективные и ресурсоемкие подходы. Это помогает сократить время, затрачиваемое на тестирование, и повышает общее качество разработки веб-приложений. Таким образом, использование алгоритмов машинного обучения при тестировании веб-приложений является эффективным и перспективным подходом, который помогает повысить точность и оперативность тестирования web-приложений.

Одним из ключевых аспектов этого преимущества является возможность автоматической адаптации алгоритмов к изменениям в

приложении, таким как добавление новой функциональности, изменение интерфейса или оптимизация кода.

Алгоритмы машинного обучения могут обновляться и реконфигурироваться в режиме реального времени на основе новых данных и событий, что позволяет им оставаться актуальными и эффективными в динамичной среде разработки. Это особенно важно в условиях быстрого развития и меняющихся требований к web-приложениям, где необходимо быстро реагировать на изменения и обеспечивать высокую степень надежности и функциональности [40]. Адаптивность алгоритмов машинного обучения позволяет им эффективно работать в различных условиях и сценариях использования приложений. Они способны адаптироваться к различным типам данных, структурам приложений и требованиям к тестированию, что делает их универсальным и гибким инструментом для автоматизации тестирования web-приложений.

Еще одним преимуществом является высокая степени охвата, обеспечиваемой алгоритмами машинного обучения при тестировании веб-приложений, важно для обеспечения полноты и глубины тестирования. Это преимущество проявляется в нескольких аспектах. Алгоритмы машинного обучения способны обрабатывать большие объемы данных и анализировать различные аспекты web-приложений, что позволяет охватить широкий спектр вариантов использования и возможных проблем. Они могут выявлять различные типы дефектов и ошибок, включая те, которые могут быть упущены при ручном тестировании. Также алгоритмы машинного обучения могут автоматически создавать и адаптировать тестовые сценарии на основе обнаруженных закономерностей и зависимостей в данных. Это позволяет эффективно исследовать различные аспекты функциональности приложения и выявлять потенциальные уязвимости или неожиданное поведение. Более того, алгоритмы машинного обучения могут учитывать и анализировать различные факторы, включая внешние и изменения в среде приложения. Это позволяет

создавать более реалистичные и разнообразные сценарии тестирования, отражающие реальные условия использования приложения.

Но, как и всех технологий есть и недостатки, перейдем к их перечислению. Недостаток, связанный с необходимостью получениябольшого объема данных для эффективной работы алгоритмов машинного обучения при тестировании web-приложений, имеет несколько аспектов.

Во-первых, сбор больших объемов данных может потребовать значительных усилий и ресурсов. Если приложение находится на начальной стадии разработки или имеет небольшую базу пользователей, получение достаточного количества данных может оказаться сложной задачей. Это может привести к задержкам в начале процесса тестирования и замедлить разработку. Также, не всегда возможно получить данные, которые полностью охватывают все возможные сценарии использования приложения. web-приложения могут иметь множество вариантов использования и различных рабочих сценариев, и для достижения полного охвата тестированием может потребоваться огромное количество данных. В таких случаях недостаток данных может привести к недостаточной эффективности и точности тестирования. Другим недостатком является необходимость обработки и анализа больших объемов данных, что может потребовать значительных вычислительных ресурсов и времени. Обучение моделей машинного обучения на больших объемах данных может быть ресурсоемким процессом, особенно если требуются сложные алгоритмы или большое количество функций.

Недостаток, связанный со сложностью настройки алгоритмов машинного обучения в контексте тестирования web-приложений, включает в себя несколько аспектов. Эффективная работа алгоритмов машинного обучения требует глубокого понимания принципов их функционирования со стороны специалистов. Это включает в себя знание математических и статистических концепций, а также опыт работы с различными типами алгоритмов и моделей. Сложность настройки связана с тем, что для

достижения оптимальных результатов необходимо правильно подобрать и настроить параметры алгоритмов, что требует определенного опыта и специальных знаний. Также, выбор подходящего алгоритма машинного обучения и его параметров может зависеть от конкретных характеристик и особенностей web-приложения, таких как тип данных, объем данных, специфика задач тестирования и т.д. Это означает, что настройка алгоритмов машинного обучения может потребовать индивидуального подхода и адаптации к конкретному контексту приложения. Другим аспектом сложности является необходимость проведения экспериментов и оценки эффективности различных моделей и параметров. Это может потребовать значительного количества времени и ресурсов, особенно если вам нужно рассмотреть множество альтернативных вариантов и сравнить их эффективность.

Еще одним недостатком является сложность интерпретациирезультатов. Результаты, полученные с помощью алгоритмов машинного обучения, могут быть трудными для понимания и интерпретации, особенно для людей без специального образования или опыта в машинном обучении. Это связано с тем, что алгоритмы могут использовать сложные математические модели и алгоритмизации, а также работать с большими объемами данных, что затрудняет понимание их результатов неспециалистами. Кроме того, результаты работы алгоритмов машинного обучения могут быть неоднозначными и требовать дополнительного анализа и оценки. Например, модель может выявить некоторые закономерности или зависимости в данных, которые могут быть непонятны или непредсказуемы с точки зрения человеческого понимания. Это может создать трудности при интерпретации результатов и принятии решений на их основе. Важно также отметить, что результаты работы алгоритмов машинного обучения могут быть взаимосвязаны с различными факторами и контекстом применения, что усложняет их интерпретацию. Например, результаты тестирования могут зависеть от специфики данных, используемых для обучения моделей, а также

от условий и параметров, в которых проводится тестирование [33]. В таблице 2 рассмотрены основные преимущества и недостатки применения алгоритмов машинного обучения для тестирования. И исходя из результатов, преимуществ оказывается больше и для исследования, которое связано с увеличением качества и скорости работы тестирования путем его автоматизации осиновым преимуществом является повышение скорости тестирования и точности результатов после его проведения.

Таблица 2 - Преимущества и недостатки применения алгоритмов

Преимущества	**Недостатки**
Повышенная точность и эффективность тестирования	Сложность настройки алгоритмов
Способность к адаптации к изменяющимся условиям	Необходимость большого объема данных для обучения
Высокая степень покрытия различных сценариев	Необходимость в интерпретации результатов
Повышение скорости тестирования	—

В контексте тестирования web-приложений алгоритмы машинного обучения имеют как преимущества, так и недостатки. К преимуществам относятся повышенная точность и эффективность тестирования, способность адаптироваться к изменяющимся условиям, а также высокая степень охвата различных вариантов использования приложений. Однако присутствуют и недостатки: сложность настройки алгоритмов, необходимость в большом объеме данных для обучения, а также требование к интерпретации результатов, что может быть затруднено из-за специфики алгоритмов и сложности получаемых данных. В целом, несмотря на потенциальные трудности и ограничения, использование алгоритмов машинного обучения при тестировании веб-приложений является перспективным подходом, который может повысить эффективность и качество процесса тестирования.

Однако важно учитывать как их преимущества, так и недостатки, чтобы оптимально использовать их в практике разработки.

В рамках первой главы было рассмотрено множество источников литературы, являющихся опорными для данной работы. Была описана теоретическая основа видов тестирования программного обеспечения, в частности веб-приложений и их особенности. В ходе теоретического анализа также были рассмотрены особенности автоматизированного тестирования, что является основой для применения машинного обучения.

Далее, опираясь на результаты различных исследований, были описаны задачи, которые могут быть решены с помощью искусственного в области тестирования, которые в дальнейшем будут проанализированы более детально и реализованы на практике.

2 Разработка и выбор алгоритма машинного обучения для автоматизации тестирования веб-приложений

2.1 Эмпирический анализ алгоритмов ML для тестирования веб приложений

В ходе теоретического обзора в предыдущей главе были затронуты основные алгоритмы машинного обучения, которые применяются тестировании web-приложений, а также были показаны основные преимущества и недостатки каждого из алгоритмов. Для более подробного изучения алгоритмов и применения на практике рассмотрим каждый из них уже не только в теоретическом плане, а работу непосредственного каждого алгоритма с технической точки зрения. Для начала и более подробного анализа алгоритмов стоит понимать, что такое машинное обучение и как оно устроено.

Машинное обучение (англ. machine learning) – обширный подраздел искусственного интеллекта, математическая дисциплина, использующая разделы математической статистики, численных методов оптимизации, теории вероятностей, дискретного анализа и извлекающая закономерности из данных. Машинное обучение уже нашло применение в следующих областях: в биоинформатике, в медицине (медицинская диагностика), в геологии и геофизике, в социологии, в экономике (кредитный скоринг, предсказание оттока клиентов, обнаружение мошенничества, биржевой технический анализ, биржевой надзор), в технике (техническая диагностика, робототехника, компьютерное зрение, распознавание речи), в офисной автоматизации (распознавание текста, обнаружение спама, категоризация документов, распознавание рукописного ввода). Сфера применений машинного обучения постоянно расширяется. Повсеместная информатизация приводит накоплению огромных объемов данных в науке, производстве, бизнесе,

транспорте, здравоохранении. Возникающие при этом задачи прогнозирования, управления и принятия решений часто сводятся к обучению по прецедентам.

Системы с машинным обучением предполагают, что программа, которая оснащена определённым алгоритмом проводит предварительную обработку данных, а затем “обучается” для выполнения определенной задачи, исходя из цели, для которой был он, был подключен. Такие системы помогают лучше распознавать тексты, изображения и голосовые команды [19].

Для решения задач в области искусственного интеллекта с помощью машинного обучения выделают несколько основных категорий в зависимости от характера данных и требуемое результата. Рассмотрим каждую задачу для понимания и выбора, наиболее подходящего для тестирования web-приложений.

Классификация: Разделение объектов на определенные классы на основе их атрибутов.

Примеры: классификация электронной почты как спама или не спама, классификация изображения и т. д.

Регрессия: Прогнозирование непрерывного значения для заданного набора атрибутов.

Примеры: прогнозирование цены на недвижимость, оценка спроса на товары и т. д.

Кластеризация: Группировка объектов на основе их сходства без заранее определенных меток классов.

Примеры: сегментирование клиентов для маркетинговых стратегий, выявление закономерностей в данных и т. д.

Обучение без учителя:

– Поиск скрытых закономерностей в данных без явного обучения на основе меток классов.

– Примеры: поиск аномалий в данных, снижение размерности и т. д.

– Обучение с подкреплением:

– Обучение агента взаимодействию с окружающей средой с целью максимизации некоторого вознаграждения.

– Примеры: управление роботами, игры и т. д.

– Обучение на основе текста и последовательности:

– Анализ и обработка текстовых данных и последовательностей.

– Примеры: машинный перевод, анализ тональности текста и т. д.

– Обучение на основе изображений и видео:

– Обработка и анализ изображений и видео.

– Примеры: классификация изображений, обнаружение объектов и т.д.

После перечисления основных категорий машинного обучения стоит разобрать более подробно каждое из них.

Классификации – это определение принадлежности объекта наблюдения к определенному классу. При наличии множества объектов, у которых уже известна принадлежность к классам, можно построить алгоритм, который сможет определять, к какому из классов будет принадлежать новый объект. В терминах машинного обучения объекты с известной принадлежностью к классам называются обучающей выборкой, а задача определения принадлежности нового объекта к классу – классификацией. Алгоритм, с помощью которого определяется принадлежность к классу, – классификатор. Одним из наиболее распространенных примеров классификации является задача выявления спама в электронной почте. В этом случае каждое письмо можно разделить на два класса: спам или не спам. Модель машинного обучения обучается на основе атрибутов в представленных письмах, таких как наличие определенных ключевых слов или фраз, и использует эти атрибуты для предсказания класса новых писем. Другой базовый пример - классификация изображений, когда модель должна определить, к какой категории относится объект, основываясь на его изображении. Например,

задача классификации изображений животных может включать в себя определение того, является ли изображение домашнимживотным, например, кошкой, собакой или другим животным. Для этогомодель обучается на наборе изображений с соответствующими метками классов и использует извлеченные из них признаки для классификации новыхизображений.

Классификация также может применяться в тестировании как процесс отнесения различных объектов или событий тестирования к определенным категориям или классам на основе их характеристик или атрибутов. Этот метод машинного обучения позволяет автоматически обрабатывать результаты тестирования и принимать решения на основе обнаруженных дефектов или аномалий.

Таким образом, классификация — это мощный инструмент машинного обучения, который позволяет автоматически относить объекты к определенным категориям на основе их характеристик, что делает ее важной задачей в различных областях исследований и приложений.

Следующей категорией является регрессия - задача машинного обучения, которая направленная на прогнозирование величины на основе набора признаков или переменных. Эта задача имеет широкий спектр применения в различных областях, включая экономику, финансы, медицину, инженерное дело и многое другое. В контексте задач машинного обучения регрессия обычно означает построение модели, которая может предсказывать числовые значения на основе предоставленной базы данных.

Один из наиболее базовых примеров регрессии - прогнозирование цен на недвижимость. В этом случае модель обучается на основе таких характеристик недвижимости, как размер, количество комнат, местоположение и т. д., и использует эти атрибуты для прогнозирования стоимости новых объектов [49]. Так же примером тестировании web-приложений связан с оценкой производительности и нагрузочным тестированием. Регрессионный анализ может использоваться для

прогнозирования производительности web - приложения при изменении нагрузки пользователей на сайте или набора запросов так же связанным с нагрузкой трафика.

Таким образом, регрессия — это один из основных инструментов машинного обучения, который позволяет предсказывать значения непрерывных переменных на основе имеющихся данных. Эта задача имеет широкий спектр применения и играет важную роль в анализе данных и в итоговые результаты в них.

Так же из основных категорий по решению задач машинного обучения можно назвать кластеризацию, которая используется для объединения схожих объектов в наборе данных в различные кластеры или группы без предопределенных меток классов. Этот метод позволяет выявить скрытые структуры в данных и обнаружить внутренние закономерности, не требуя предварительного разделения данных.

В контексте машинного обучения кластеризация часто применяется для решения таких задач, как сегментация клиентов, анализ социальных сетей, обнаружение аномалий и категоризация данных. Например, в маркетинге кластеризация может использоваться для выделения различных групп клиентов на основе их предпочтений, покупательского поведения и характеристик. Это позволяет компаниям создавать более целевые маркетинговые стратегии и персонализированные подходы к обслуживанию клиентов.

Так же еще одним из примеров кластеризации является метод k-средних, который делит данные на k кластеров, где каждый кластер характеризуется средним значением объектов внутри него. Другой метод - иерархическая кластеризация, которая строит дерево кластеров на основе иерархии сходств между объектами.

В тестировании примером использования кластеризации может быть анализ журналов сервера. Справиться с большим объем журналов сервера,

содержащих информацию о запросах к web-приложению, таких как URL, IP-адреса клиентов, типы запросов и т. д. Задача состоит в том, чтобы выявить в этих данных скрытые паттерны и кластеры, которые могут указывать на различные аспекты работы приложения или поведения пользователей. Использование алгоритмов кластеризации позволяет автоматически группировать похожие запросы или события в кластеры на основе их характеристик. Например, это может помочь выявить кластеры частых запросов с определенных IP-адресов, кластеры запросов, вызывающихошибки сервера, или кластеры аномального поведения, например атак наприложение. Анализ таких кластеров может дать ценную информацию для тестирования web-приложений.

Таким образом, кластеризация является мощным инструментом для анализа данных и выявления внутренней структуры, что делает ее важным элементом машинного обучения и его применения в различных областях анализа, разработки и тестирования.

Также важнейшей категорией является такие модели, которые обучаются на текстовых данных или последовательностях символов для выполнения различных задач, таких как классификация, генерация текстов, машинный перевод и другие. Одним из ключевых подходов к обучению на основе текста является использование рекуррентных нейронных сетей (RNN) и их модификаций, таких как Long Short-Term Memory (LSTM) и Gated Recurrent Unit (GRU). Эти модели способны учитывать последовательность данных и улавливать зависимости между словами или символами в тексте. RNN и их разновидности широко используются в задачах обработки естественного языка (NLP), таких как анализ тональности текста, классификация документов, генерация текста и другие [28].

Для обучения на последовательностях, не обязательно текстовых, например, временных рядах или последовательностях действий в играх, также применяются различные модели машинного обучения. Среди них

рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN), а также их комбинации, которые позволяют учитывать последовательный характер данных и выявлять в них закономерности.

Таким образом, основные категории для решения задач с помощью машинного обучения могут быть использованы для тестирования web-приложений и для дальнейшей разработки автоматизации тестирования с непосредственно применением алгоритмов машинного обучения. Каждая из этих задач имеет свои особенности и применение в различных областях. Вместе они образуют мощный инструментарий для анализа и работы с данными, что позволяет решать разнообразные задачи.

2.2 Описание и обоснование выбора алгоритма для автоматизациитестирование web -приложений

После проведенного анализа алгоритмов, задач и категорий машинного обучения, которые связаны не только с разработкой систем для продуктов, но и с тестированием web-приложений, которые помогут автоматизировать этот процесс и принести пользу компаниям и сотрудникам, работающих в них.

Важно выбрать наиболее подходящие алгоритм в контексте тестирования, разработать его и протестировать с реальными данными и оценить пользу от автоматизации.

Всего было проанализировано 5 алгоритмов:

– использование инструментов для записи и воспроизведения действий пользователя;

– использование компьютерного зрения для обнаружения элементов интерфейса;

– использование алгоритмов машинного обучения для создания тестовых данных;

– распознавание и анализ текста;

– прогнозирование критических областей.

А также категории, которые решают задачи, связанные с машинным обучением:

– регрессия;

– кластеризация;

– обучение без учителя;

– обучение с подкреплением;

– обучение на основе текстов и последовательностей;

– обучение на основе изображений и видео.

Выделим несколько алгоритмов, которые могут быть использованы на практике и помочь сотрудником компании.

Автоматическая генерация сценариев тестирования – будет являться прекрасным помощником для тестировщиков, так как написание документации является рутинным и достаточно долгим процессом, но в то же время обязательным в любом цикле разработки со стороны отдела тестирования. Так как по этой документации проверяется система вовремя тестирования отдельных элементов системы, так и во время проведения регрессионного тестирования [20]. Тестовый случай (Test Case), он же тестовый сценарий, включает в себя. Минимальный элемент тестирования (всего одна верификация\проверка); совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части; описание определенных действий и условий, которые необходимы длятого, чтобы выявить тот или иной баг.

Под тест кейсом понимается структура вида, в таблице 3 показан пример структуры тест кейса.

Таблица 3 – Пример Тест кейса

Action	Expected Result	Test Result.

Виды Тестовых Случаев Тест кейсы разделяются по ожидаемому результату на позитивные и негативные:

– позитивный тест кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию;

– негативный тест кейс оперирует как корректными, так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.

Существует множество видов оформления тест кейсов, но разберём наиболее чаще всего используемых в различных компаниях среди тестировщиков первый – в виде таблицы, второй – в виде текста. В примерах 1 и 2 покрытие будет одинаковым, но время, которое потребуется для чтобы пройти через него, будет разным.

Разберем первый случай. Для наглядного примера, как выглядит тестовая документация в виде тест кейса выглядит стандартно, как таблица.

Таблица 4 - Пример запалённой таблицы тест кейса

Проверка аутентификации		
Действие	Ожидаемый результат	Результат
Открыть систему	Отражаются элементы - Поле Логин - Поле Пароль - Кнопка “Вход” - Кнопка “Забыл пароль”	

Второй вид тест кейса представлен в виде текста, в зависимости от стандартов компании используется тот или иной способ оформления документации.

Пример тестового случая 2:

Название: Проверка аутентификации

Действия: Откройте страницу «Вход в систему».

Тест: убедитесь, что отображаемая страница соответствует странице (скриншот).

Выбор алгоритма, с помощью которого можно генерировать тестовые сценарии сделан, потому что это существенно автоматизирует работу тестировщика, так как этап написания тестовой документации будет автоматизирован и сократит время, которое бы затратил сотрудник на эту работу. Из ключевых преимуществ данного алгоритма стоит выделить. Увеличение охвата тестирования, которое позволит автоматически генерировать тестовые сценария, а именно тест кейсы, так же позволит

расширить охват тестирования, охватывая больше возможных сценариев использования приложения, чем это было возможно с использованием ручныхметодов. Это особенно важно для web-приложений, которые имеют множество функций и возможностей, и требуют обширного тестирования.

А также благодаря эффективному обнаружению дефектов. Алгоритмы машинного обучения могут анализировать большие объемы данных и выявлять сложные и неочевидные паттерны в работе приложения. Это позволяет выявлять потенциальные дефекты или проблемы, которые могли бы быть упущены при ручном тестировании.

Автоматическая адаптация к изменениям, так как этот алгоритмы машинного обучения, а именно генерация тестовых сценариев может адаптироваться к изменениям в приложении или окружающей среде, так как генетические алгоритмы или нейронные сети позволяют создавать более устойчивые и адаптивные тестовые сценарии, которые остаются актуальными даже при изменении функциональности приложения.

Так же, если затрагивать тему времени и концепцию время равно деньги

особенно в крупных проектах и компаниях, то экономия времени и ресурсов-является важнейшим инструментом. Автоматическая генерация тестовых сценариев может существенно сократить время, затрачиваемое на тестирование, и уменьшить необходимость в человеческих ресурсах. Это особенно важно для крупных проектов, где ручное тестирование может быть трудоемким и затратным процессом.

Улучшение качества тестирования этот критерий показывает, что тест кейсы, созданные вручную являются менее полными и комплексными, чем те, которые разрабатываются искусственным интеллектом [17]. Это позволяет выявлять более широкий спектр проблем и дефектов в приложении, что в итоге способствует улучшению его качества и надежности.

Таким образом, выбор алгоритма машинного обучения для автоматической генерации тестовых сценариев в тестировании web-приложений обоснован необходимостью повышения эффективности, охвата и качества тестирования, что в итоге способствует повышению уровня надежности и удовлетворенности пользователей.

При генерации тестовых данных необходим критерий останова процесса генерации тестовых данных, критерий адекватности тестовых данных, иначе система генерации тестовых данных будет генерировать данные бесконечно долго. В качестве такого критерия может использоваться некоторая метрика, специфичная для тестируемой программы и выбранной методики тестирования. Таким образом, задача генерации тестовых данных сводится к задаче математической оптимизации целевой функции, которая является математически формализованным критерием адекватности для тестируемой программы. Алгоритмы генерации тестовых данных в своей работе используют различные алгоритмы математической оптимизации для достижения оптимума на заданной целевой функции.

Также необходим критерий покрытия для определения качества получившегося теста или набора тестов. Такой критерий показывает в

процентном соотношении, насколько тест или набор тестов удовлетворяют установленным требованиям.

Классификация критериев покрытия:

1) Критерии покрытия типа «черный ящик»
2) Критерии, основанные на анализе спецификаций
3) Критерии, основанные на анализе UML-диаграмм
4) Критерии покрытия типа «белый ящик»
5) Критерии потока управления
6) Критерии потока данных

Все эти критерии применяются в тестировании web-приложений.

2.3 Построение модели с помощью цепей Маркова

Основой для генерации тестовых сценариев, а именно тест кейсов можно выбрать теорию цепей Маркова. Для этого надо разобраться что это такое?

Цепь Маркова – это последовательность случайных событий с конечным или счётным числом исходов, где вероятность наступления каждого события зависит только от состояния, достигнутого в предыдущем событии [38]. Характеризуется тем свойством, что, говоря нестрого, при текущем настоящем состоянии системы, её будущее состояние не зависит от прошлого. Иными словами цепь Маркова — это прежде всего инструмент из теории случайных процессов, состоящий из последовательности n количества состояний. Связи между узлами (значениями) цепочки при этом создаются, только если состояния стоят строго рядом друг с другом. Данная теория отлично подходит для введения алгоритма машинного обучения для автоматизации тестирования. Так как генерация тестовых сценариев будет происходит автоматически и сократит время и силы сотрудникам компании.

Для начала нужно понять, как работает теория цепей Маркова без

применения машинного обучения. Самым базовым примером является предложение из детской книги, благодаря которой видно базовую концепцию цепи Маркова, а также будет понятным определение, что такое в контексте цепей Маркова корпус, звенья, распределение вероятностей и гистограммы.

Автоматическая адаптация к изменениям, так как этот алгоритмы машинного обучения, а именно генерация тестовых сценариев может адаптироваться к изменениям в приложении или окружающей среде, так как генетические алгоритмы или нейронные сети позволяют создавать более устойчивые и адаптивные тестовые сценарии, которые остаются актуальными даже при изменении функциональности приложения.

Так же, если затрагивать тему времени и концепцию время равно деньги особенно в крупных проектах и компаниях, то экономия времени и ресурсов-является важнейшим инструментом.

Математическое пояснению данной теории. Многие современнее методы обработки естественного языка существенно зависят от данных, которые будут использоваться для обучения модели. Для обучения моделей машинного обучения требуется значительный объем данных, а в некоторых сценариях данных может оказаться недостаточно. Для того, чтобы отразить

Последовательность дискретных случайных величин $\{X_n\}$, $n \geq 0$ будем называть простой цепью Маркова.

$$P(X_{n+1} = i_{n+1} \mid X_{n=i_n}, X_{n-1} = i_{n-1,\ldots,}, X_0 = i_0) = P(X_{n+1} = i_{n+1} | X_n = i_n), \quad (1)$$

где, $P_{ij}(n) = P(X_{n+1} = j \mid X_n = i)$ переходная матрицей вероятностей на n-ом шаге.

Формула (1) задает условное распределение последующего состояния цепи Маркова, которое зависит только от текущего состояния и не зависит от всех предыдущих состояний. При этом $\{X_n\}$ называют пространством

состояний цепи, а n – номером шага., а вектор $p = (p_1, p_2, \ldots.)$ – начальным распределением цепи Маркова. Матрица $P_{ij}(n)$ является стохастической и для нее выполнятся условие нормировки [43].

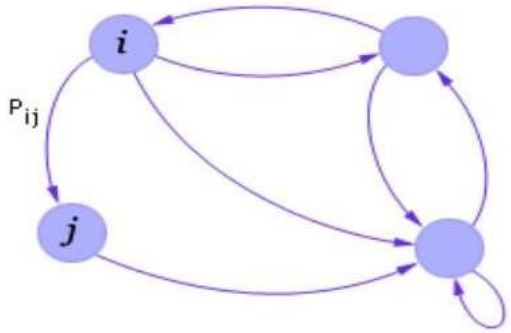


Рисунок 2 — Цепь Маркова с четырьмя состояниями

Переходная матрица состояния — это фундаментальная матрица решений векторного линейного однородного дифференциального уравнения с переменными коэффициентами.

Она позволяет перейти из начального состояния в любое другое в пространстве состояний. Переходная матрица и однородные цепи выглядят так:

$$P(n), \text{ где } P_{ij}(n) = P(X_{n+1} = j \mid X_n = i, \qquad (2)$$

где, $p_i = P(X_0 = i)$ – начальным распределением цепи Маркова.

Матрицапереходных вероятностей является стохастической справа, то есть цепь Маркова называется однородной, если матрица переходных вероятностей не зависит от номера шага, то есть:$P_{ij}(n) = P_{ij}$, в противном случае цепь Маркова называется неоднородной. В дальнейшем будем предполагать, что имеем дело с однородными цепями Маркова.

Алгоритмом для реализации генерации тестовых сценариев с помощью

базы данных тестовых случаев и обучения нейронных сетей с цепами Маркова выглядит так, первый этап — это подготовка данных.

Нужно загрузить данные из базы данных тестовых случаев для обучения модели. Подготовка данных, включая предварительную обработку, очистку и представление в удобном для обучения формате. Изначально тест кейсы выглядят в виде множества таблиц, которые нужно сформировать в одну большую таблицу со всеми данными.

Набор данных, представленный в виде таблицы. Обозначим эту таблицу как D, где каждая строка соответствует отдельному тестовому сценарию, а каждый столбец - признаку или шагу теста. Предположим, есть N тестовых сценариев и M атрибутов в каждом. Далее необходимо разделить данные. Разделим набор данных D на матрицу признаков X и вектор целевых переменных y. Матрица признаков X будет иметь размерность N × M, где каждая строка представляет собой признаки для отдельного тестового сценария. Вектор целевых переменных y будет содержать метки целевых классов (например, пройден или не пройден тест) для каждого тестового сценария.

Далее идет обучение модели машинного обучения. Подходящую модель машинного обучения для обработки тестовых случаев (например, классификацию, генеративные модели и т. д.). В качестве модели можно использовать модель многослойного перцептрона (MLP). Данную модель можно описать следующим образом для пояснения с точки зрения математической модели:

$$z^{(1)}=XW^{(1)} + b^{(1)}a^{(1)}=\sigma(z^{(1)})z^{(2)}=a^{(1)}W^{(2)} + b^{(2)}, \quad (3)$$

где X- матрица признаков$W^{(1)}$;

$W^{(2)}$ – матрица весов;

$b^{(1)}b^{(2)}$- векторы смещения;

σ- функция активности;

$z^{(1)}$, $z^{(2)}$ – линейные комбинации выходных данных и весов.

После обучения модели можно проверить ее на точность для этого можно использовать функцию потерь, например, среднеквадратичную ошибку (MSE) или кросс-энтропию. Это математическое описание процесса обучения модели машинного обучения на основе данных из базы данных тестовых сценариев с использованием многослойного перцептрона [26].

Конечно, у каждого алгоритма есть особенности и, поэтому стоит разобрать особенности применения данного метода в контексте генерации тестовых сценариев с применением машинного обучения. Далее можно совместить машинное обучение для генерации тестовых сценариев и цепи Маркова. Для создания модели необходимо понимать шаги, которые будут использованы для создания модели.

Первым этапом является подготовка базы данных для обучения и последующей генераций тестовых сценариев она может включать в себя записи, которые были созданы вручную командой тестирования при реализации прошлых новых функциональности приложения, состояний

приложения, вводимых данных и ожидаемых результатов для каждого тестового случая по стандартному шаблону создания тест кейса.

Вторым этапом идет обучение модели на базе данных, которая показывает, как написаны тест кейсы. На этих данных необходимо обучить модель машинного обучения. В зависимости от постановки задачи это регрессионная модель, которая предсказывает последующие действия пользователя или состояния приложения на основе предыдущих данных.

Следующим этапом является разработка модели цепи Маркова. Здесь необходимо разработать модель, которая будет использоваться для генерации последовательностей действий на основе состояний, предсказанных моделью машинного обучения. Эта модель должна учитывать вероятности переходов

между состояниями и генерировать последующие состояния в соответствии с этими вероятностями.

Далее идет генерация тестового сценария. После обучения моделей и разработки модели цепи Маркова можно приступать к генерации тестовых сценариев. Для этого сначала используется модель машинного обучения для предсказания последующих состояний или действий на основе текущего состояния или действия. Затем модель цепи Маркова используется для генерации последующих состояний или действий на основе предсказанных вероятностей перехода.

Последним этапом является оценка и анализ сгенерированных сценариев. После генерации тестовых сценариев необходимо оценить их качество и пригодность для тестирования веб-приложений. Это может включать анализ покрытия функциональности приложения, разнообразия сценариев и их практической применимости. Полученные результаты могут быть использованы для улучшения моделей и методов генерации тестовых сценариев.

Модель генерации тестовых примеров на основе цепи Маркова с использованием методов машинного обучения имеет различные особенности. В основе которых лежат стандартные особенности обучения модели с использованием машинного обучения. Необходимость качественных данных, сложность моделирования, переобучение модели, необходимость обновления модели.

Качество базы данных, по которой будет обучаться модель, является важнейшей частью обучением, в общем и целом, так как корректность генерируемых тестовых сценариев зависит от качества данных в базе данных тестовых сценариев. Недостаточное количество данных или ненадежные данные могут привести к неправильным прогнозам модели. Сложность моделирования, а именно генерация тестовых сценариев так же является особенностью построения модели, так как динамика взаимодействия

пользователя с веб-приложением с помощью цепи Маркова может быть сложной из-за большого количества возможных состояний и действий пользователя. Для отладки данной особенности необходимо хорошо выбирать размерность состояний и учитывать действия пользователей, которые опять же могут быть учтены в базе данных.

Возможно, самое распространённая особенность обучения машинного обучения является переобучение модели — модель слишком хорошо адаптируется к обучающим данным, изучая свои шумы и особенности, а не обобщая их на общие закономерности. Переобучение может привести к тому, что модель показывает высокую точность на обучающем наборе данных, но плохо обобщается на новые, ранее не встречавшиеся данные. При создания тестовых сценариев для тестирования веб-приложений переобучение модели может проявляться в разных случаях. Например, слишком точное моделирование конкретных сценариев, модель может перетренироваться на определенных действиях из базы данных, запомнив порядок действий и состояний приложения в обучающих примерах. В результате она может генерировать сценарии, которые слишком точно соответствуют обучающим данным, но не учитывают возможные вариации и аномалии, которые могут возникнуть при реальном использовании приложения. Необходимость обновления модели, так же является особенностью модели в контексте генерации тестовых сценариев, так как они могут меняться, исходя из появления в приложении новых возможностей и функционала доступного для пользователя. Поэтому важно постоянно обновлять обучающие данные для поддержания актуальности модели и ее корректной работы, чтобы она действительно приносила пользу тестировщику.

В рамках второй главы было выполнено описание предметной области и рассмотрены модели машинного обучения, применяемые для тестирования web-приложений. Выявлены основные функциональные требования для разрабатываемой системы для генерации тестовых сценариев с помощью

цепей Маркова, выявлены риски и особенности для выполнения задачи. Также спроектированы шаги для разработки системы, которые будут выполнены в дальнейшем.

По результатам работы во второй главе можно приступать к разработке системы.

3 Разработка решения и оценка продуктовой эффективности

3.1 Подготовка данных и анализ входной информации

Для реализации архитектуры, в которой модель обучается на базе данных примеров тестовых сценариев, а нейросеть с использованием цепей Маркова и машинного обучения генерирует тестовые сценарии для тестирования веб-приложений, предлагается следующая структура:

Для реализации этого метода необходимо использовать базу данных тестовых примеров, на которых модель будет обучаться. Базу данных из примеров тест-кейсов в виде множества

База данных может быть использована для обучения нейронной сети, где шаги тестирования и ожидаемые результаты будут входными данными, а фактические результаты - целевыми значениями. На основе этого обучения нейросеть сможет генерировать новые тестовые сценарии, используя цепи Маркова и методы машинного обучения. Разбор базы данных является важнейшим этапом разработки, так как обучение модели основано на базе данных, которая состоит из тест кейсов.

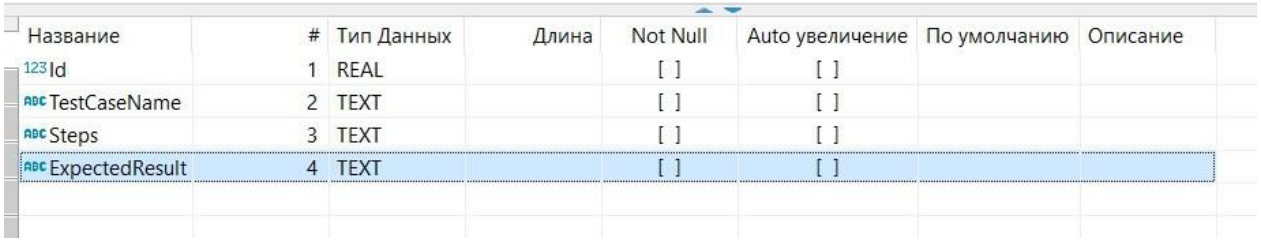
Архитектура базы данных в данном случае очень простая, для обучения модели необходима одна таблица состоящая из 4 колонок

В случае использования стандартных тест-кейсов для ручного тестирования веб-приложений как основы для базы данных будет содержать следующие элементы. Метаданные (Id).

Дополнительная информация о тест-кейсе, такая как его уникальный идентификатор, приоритет, статус выполнения, дата создания и обновления, ответственный за выполнение и т. д. Шаги тестирования (Steps). Последовательность действий, которые должны быть выполнены для выполнения конкретного тест-кейса. Например, это могут быть шаги взаимодействия с интерфейсом веб-приложения, такие как открытие

страницы, заполнение формы, нажатие кнопок и т. д. Название кейса (TestCaseName). Описание того, что ожидается увидеть. Это может быть корректное описание определенного действия, появление определенного сообщения об успешном завершении операции. Ожидаемые результаты (ExpectedResult).

Результаты, которые ожидаем увидеть после каждого шага тестирования. Если есть расхождения между ожидаемыми и фактическими результатами, они могут быть зафиксированы как потенциальные проблемы.



Название	#	Тип Данных	Длина	Not Null	Auto увеличение	По умолчанию	Описание
Id	1	REAL		[ ]	[ ]		
TestCaseName	2	TEXT		[ ]	[ ]		
Steps	3	TEXT		[ ]	[ ]		
ExpectedResult	4	TEXT		[ ]	[ ]		

Рисунок 3 - Структура базы данных

В таблице представлены типы данных двух видов: цифровые и тестовые. В колонках, которые передают данные с шагами, результатами и названием кейсов – являются тестовыми. Цифровая колонка – это порядковой номер кейса.

Сбор информации для базы данных является сложным и основополагающим в работе и обучении модели. Прежде всего была создана база данных с помощью SQLite, Python и Google colab с локальным форматом подключением. Основой для базы данных являются тест кейсы, написанные для проверки существующего web-приложения для девелоперской компании “ПИК”, с помощью этих тест кейсов проверяется основой функционал во время регрессионного тестирования системы. Все тест кейсы находятся в специализированном ПО, которое необходимо для администрирования и хранения тестовой документации.

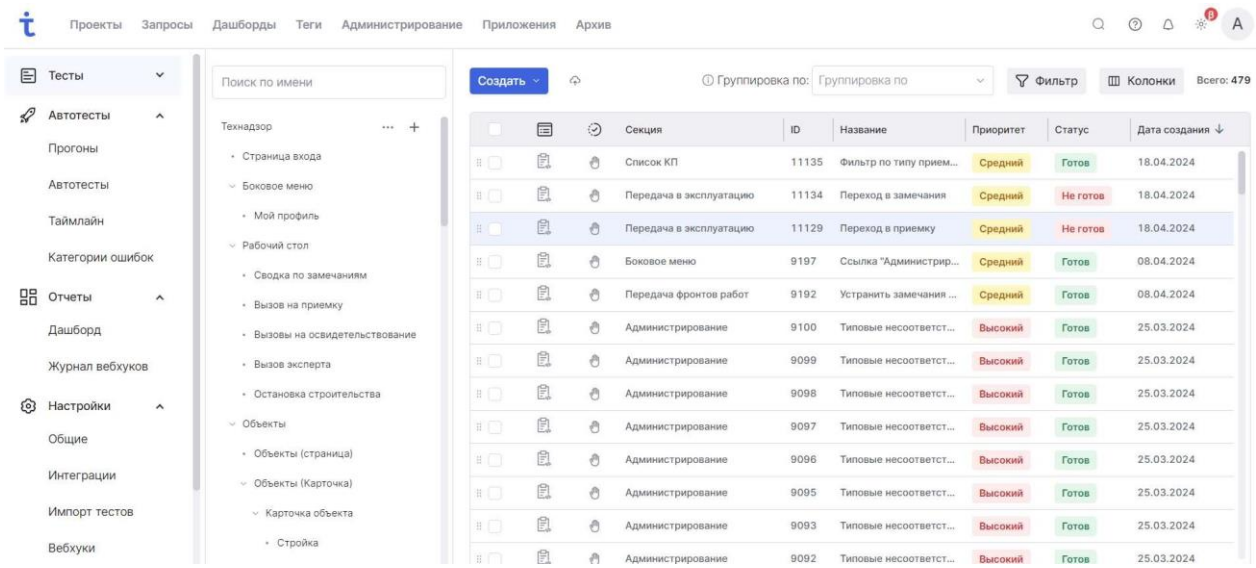


Рисунок 4 - Тест кейсы на TestIT

Общий репозиторий на отдельный проект на основе, которого будет обучаться модель представлена на рисунке 8.

Так же тест кейс оформлен по специфике ПО и содержание кейса написано по правилам языка Girkin для дальнейшей автоматизации клиентской части проекта, поэтому для начала было необходимо очистить некоторую информацию в кейсах для корректного обучения модели.

Язык Girkin для написания тест кейсов является особенным при написании тест кейсов, он состоит из слов Given, When, And, Then – эти слова даны для описания шагов воспроизведения действий в web-приложении, чтобы получить точное понятия действий и что за каким-то действием следует какой-то результат, например When (когда) Я нажимаю поле “Фильтр по типу” And (и) Я выбираю тип приемки Then (тогда) Я вижу список приемок по выбранному типу. Для обучения модели такие элементы в описании тестовых кейсов не важны, так как кейсы будут генерироваться автоматически. Разберем поподробнее, в чем же задача специалиста по автоматизации. Используя любой плагин для автоматизации, который поддерживает язык Gherkin, специалист по автоматизации тестирования берет сценарии и создает Java-файл для кода автоматизации.

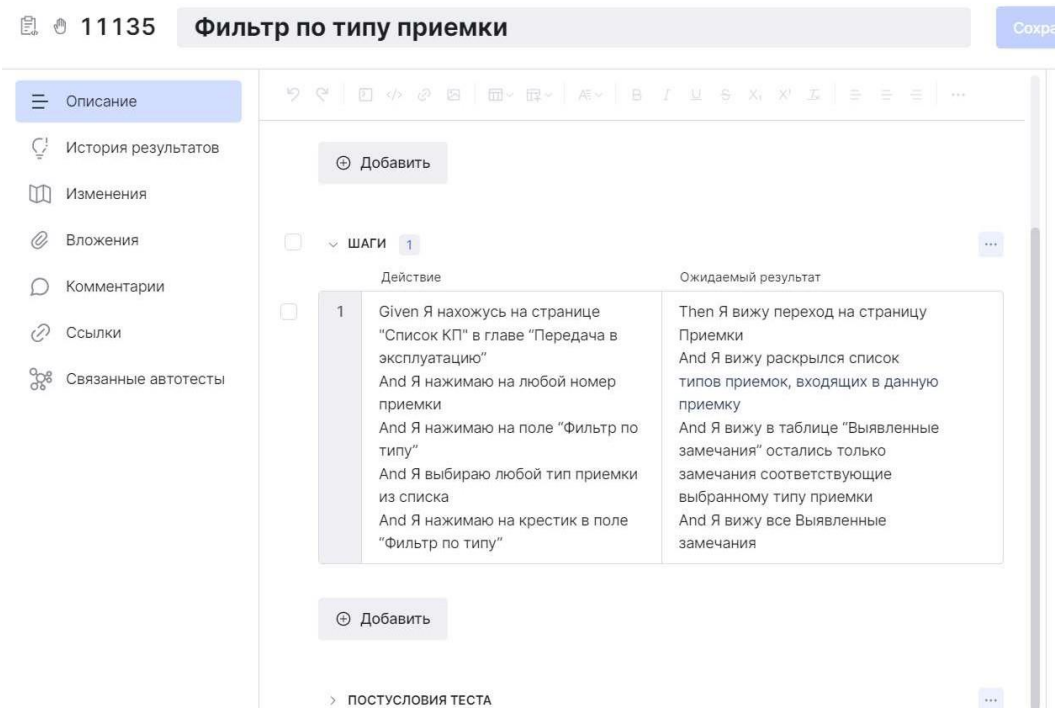


Рисунок 5 - Пример тест кейса на TestIT

Для создания Базы данных был написан код, позволяющий создать подключение для СУБД SQLite, так же была добавлена одна таблица, куда были загружены тест кейсы. Изначально данные были не подготовлены к обучению, так как после выгрузки из системы TestIT в файле были много дополнительных колонок, которые не будут учтены при обучении модели и их пришлось удалять с помощью SQL скриптов. Таким образом, данные, которые были очищены, можно использовать для обучения и применения системой автоматизированной генерации тестовых сценариев. Чтобы получить конкретные тест кейсы нужно загрузить в системы примеры написания тест кейсов, которые были уже использованы в этом разделе ранее для того, чтобы модель обучилась и могла генерировать подобные кейсы самостоятельно, что позволит тестировщику существенно сократить время на ручное написание тест кейсов. Так как это достаточно трудоемкий и время затратный процесс, но необходим для поддержания тестовой документации в порядке и покрытия системы тест кейсами.

cases | Введите SQL выражение чтобы отфильтровать результаты

Id	TestCaseName	Steps	ActualResults
1 260	Авторизация через логин и пароль	В поле "Логин" ввожу imurzaliev@test.com¶ В пол	Я на главной странице сайта¶
1 275	Выход из аккаунта	Нажимаю на кнопку "Выйти" рядом с ФИО пользо	Нажимаю на кнопку "Выйти" рядом с ФИО пользо
1 404	Смена пароля через кнопку "Я забыл свой пароль"	Нажимаю на ссылку "Я забыл свой пароль" Заг	Я вижу форму восстановления пароля
1 405	Кнопка выдвижения панели	Я нахожусь на любой странице	Я вижу, что левая панель меню содержит следующи
1 499	Ссылка "Сводка по замечаниям"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https:/dashboard
1 500	Ссылка "Вызовы на приемку"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 501	Ссылка "Вызовы на освидетельствование"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 502	Ссылка "Вызов эксперта"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
8 583	Ссылка "Остановка строительства"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 503	Ссылка "Объекты"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 507	Ссылка "Замечания на стройке"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 508	Ссылка "Замечания на отделке"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 509	Ссылка "Замечания на заселении"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 510	Ссылка "Замечания"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
8 580	Ссылка "Список КП"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
9 197	Ссылка "Администрирование"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 511	Ссылка "Список КО"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 512	Ссылка "Замечания (требуется КО)"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 514	Ссылка "Замечания (проведен КО)"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 513	Ссылка "Замечания по МОПам"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 515	Ссылка "Мои объекты"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 516	Ссылка "Список замечаний"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 517	Ссылка "Список АВН/Претензии"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
8 581	Ссылка "Разрешения"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
8 582	Ссылка "Рассылки"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 518	Ссылка "Справочник работ"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc
1 519	Ссылка "Администрирование чек-листов"	Я нахожусь на любой странице Технадзора¶ Я на	Я переадресован на страницу https://tracker.prod.doc

Рисунок 6 - Содержание базы данных с тест кейсам

После первой очистки данных таблица с тест кейсами выглядит так рисунок 6, далее при обучении модели некоторые данные так же будут очищены и разбиты на тренировочные и тестовые для более качественного обучения модели и в последствии генерации тест кейсов.

3.2 Формализация процессов разработки и тестирование моделей

Для формирования генерации тест кейсов после анализа нескольких моделей и изучения материалов оптимальной моделью оказалась Модель LSTM (Long Short-Term Memory) — это вид рекуррентной нейронной сети (RNN), специально разработанный для работы с последовательными данными и предотвращения проблемы затухающего градиента, которая часто встречается при обучении обычных RNN. Основная идея LSTM заключается в том, чтобы сохранять и обновлять внутреннее состояние сети (называемое "ячейкой памяти") на протяжении различных временных интервалов. Это

позволяет LSTM моделям сохранять информацию о длительных зависимостях в данных и избегать забывания важных сигналов [18]. Основные компоненты модели LSTM включают в себя: ячейка памяти (Memory Cell) и вентили (Gates)

Основная часть LSTM, которая хранит информацию о предыдущем состоянии сети и может быть обновлена или использована в следующем временном шаге. Обновление ячейки памяти (Cell Update)

Для обновления содержимого ячейки памяти сначала создаем новый кандидат c_t, который может быть добавлен к состоянию памяти. Это определяется входными данными x_t и внутренним состоянием предыдущего шага h_{t-1}.

$$c_t = tanh(W_{xe}x_t + W_{he}h_{t-1} + b_c), \quad (6)$$

где c_t – кандидат на обновление ячейки памяти;

tanh – гиперболическийтангенс;

W_{xe} и W_{he} – весовые матрицы для входных данных и внутреннего состояния,

b_c- смещение.

Обновление состояния памяти. Обновление памяти c_t с учетом информации из входного вентиля забывающего вентиля f_t и кантата c_t

$$c_t = f_t c_{t-1} + i_t c_t, \quad (7)$$

LSTM содержит несколько вентилей (входной, забывающий и выходной), которые регулируют поток информации в и из ячейки памяти. Эти вентили помогают модели определять, какую информацию следует сохранить или забыть. Входной вентиль решает, какая информация должна быть обновлена в ячейке памяти c_t. Он принимает входные данные $x_{t\ и}$ внутренние

ссостояния предыдущего шага h_{t-1} и возвращающий вектор значений между 0 и 1, где 1 обозначает полную релевантность для информации.

$$i_t = \sigma(W_{xi} \ x_t + \ W_{hi} h_{t-1} + b_i, \quad (8)$$

где i_t − вектор входного вентиля;

σ −сигмоидальная функция активации;

W_{xi} и W_{hi} весовые матрицы длявходных данных и внутреннего состояния;

b_i – смещение.

Забывающий вентиль (Forget Gate). Забывающий вентиль решает, какая информация должна быть забыта из предыдущей ячейки памяти c_{t-1} он принимает входные данные x_t ивнутреннее состояние предыдущего шага h_{t-1} и возвращает вектор значений между 0 и 1, где 0 означает полное забывание информации.

$$f_t = \sigma(W_{xf} \times x_t + W_{hf} \times h_{t-1} + b_f), \quad (9)$$

где f_t- вектор забывающего вентиля;

σ – сигмоидальная функция активации;

$W_{xf}x_t + W_{hf}$ – весовые матрицы для входных данных и внутреннего состояния;

b_f – смещение.

Функция активации (Activation Function). Обычно в моделях LSTM используются функции активации, такие как гиперболический тангенс и

сигмоидальная функция, для управления потокоминформации через вентили и для преобразования выходных данных.

Модели LSTM широко используются в области обработки естественного языка (Natural Language Processing, NLP), распознавания речи, временных рядов и других задач, где необходимо работать с последовательными данными и захватывать долгосрочные зависимости.

После математического обоснования и введения формул стоит перейти непосредственно к написанию кода, который будет выполнять цели по генерации тест кейсов.

Для обучения модели в коде использовались следующие библиотеки:

– sqlite3: Библиотека для работы с базами данных SQLite;

– Numpy. Библиотека для работы с многомерными массивами и математическими функциями;

– tensorflow.keras. Фреймворк глубокого обучения, включающий высокоуровневый API Keras для создания, обучения и оценки нейронных сетей;

– matplotlib.pyplot. Библиотека для визуализации данных в виде графиков.

После подключения базы данных к модели началось обучение. В ходе обучения сначала было модель была обучена на 10 эпохах, что показало неплохие результаты по показателям loss: 1.2822 - accuracy: 0.6957, но все равно недостаточно хорошими, было принято обучать модель по 5 эпох и сравнивать результаты. После обучении модели на 25 эпох получились такие графики loss и accuracy. Вывод по графикам после первой итерации обучения можно сделать такой, оба графика показывают динамику: функция потерь уменьшается с каждой эпохой, что свидетельствуетоб уменьшении ошибки модели. Точность увеличивается, что свидетельствует об улучшении предсказательной способности модели. В итоге было принято решение продолжить обучение модели для получения лучших результатов и

показания метрик.

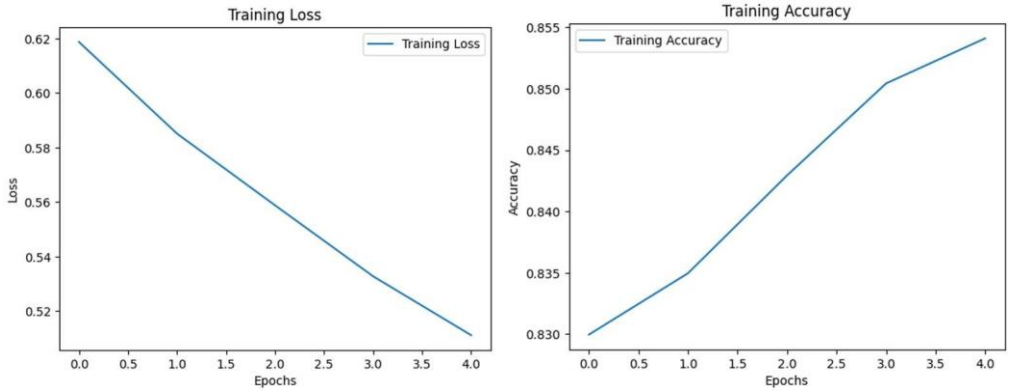


Рисунок 7 - Графики Loss и Accuracy

На графиках видно, что идет положительная динамика в ходе обучения модели, даже после 25 эпох обучения потери равны 0.62, а точность равна 0.855, что является высоким результатом при таком количестве эпох в обучении. Точность — это процент правильных предсказаний модели на тестовом наборе данных, чем выше значение точности, тем лучше модель способна классифицировать данные. Потери — это значение, которое модель стремится минимизировать в процессе обучения, чем меньше значение потерь, тем лучше модель предсказывает целевую переменную. Но для улучшения результатов было принято решение продолжать обучение.

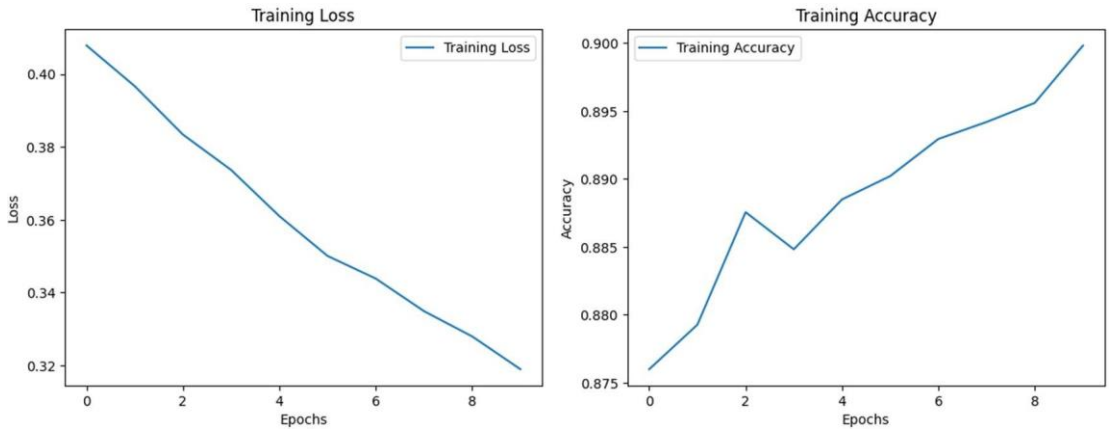


Рисунок 8 - Результаты финального обучении

Графики на рисунке 43443 сделаны после 40 эпох обучения модели, видно, что показатели улучшились по сравнению с первой итерацией обучения, но из-за всплеска на графике точности было принято решение, что стоит прекратить обучение на 40 эпохах. Результаты равны потери = 0.3190, а точность = 0.8998.

Вывод по графикам можно сделать такой, оба графика показывают положительную динамику: функция потерь уменьшается, что свидетельствует об уменьшении ошибки модели. Точность увеличивается, что свидетельствует об улучшении предсказательной способности модели. Эти результаты показывают, что модель успешно обучается и ее качество улучшается с каждой эпохой.

После успешного обучения модели нужно приступить генерации тестовых сценариев с помощью цепей Маркова на обученной модели. В ходе разработки было использовано пять функция для преобразование начального текста в последовательность токенов и соответственно работу с ними. Далее идет функция для предсказания следующего слова и обновления начального текста. После использования этих функций идут три функции непосредственно для генерации нового текста и отражения тестового сценария. Этот код позволяет на основе случайного начального текста из базы данных генерировать новый тестовый сценарий, используя цепи Маркова и модель LSTM. Он применяет обученную модель для последовательного предсказания следующих слов, что позволяет создать новый, логически связанный текст для тестирования веб-приложений.

После обучение модели и запуска кода с генерацией тестовых сценариев модель выдает хороший тест кейс, в котором описаны шаги для проверки системы. Каждая попытка генерации тест кейса показывает хороший результат, а хороший результат – это правильный порядок шагов, которые при прохождении теста кейса будет показать результат на веб сайте. Шаги идут друг за другом и не нарушают логику действий пользователя. Также видно,

что все шаги кейса начинаются с “Я”, что показывает хорошо обученную модель, так как каждый шаг связан с действием пользователя.

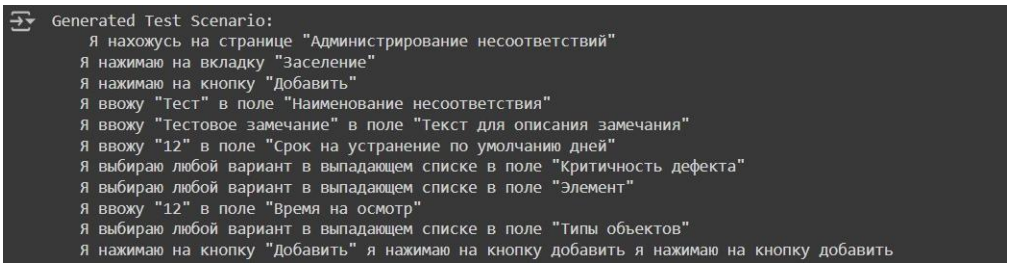


Рисунок 13 - Генерация тест кейса

Сгенерированные шаги для теста кейса существенно помогут тестировщику автоматически покрыть большим количеством тест кейсов функционал web-приложения, тем самым обеспечив проекту качество и снизить количество дефектов у пользователей, так как большое покрытие тест кейсов способствует проверке системы на стадии тестирования.

После отработки алгоритма можно увидеть, что шаги сгенерированные автоматически показывают отличный результат, так как видно, что соблюдается логика в описанных шагах, каждое действие сделают за другим и нет опечаток и ошибок.

3.3 Оценка продуктовой эффективности модели для тестирования web- приложений

Для оценки продуктовой эффективности модели для генерируемых тест кейсов для web-приложений можно обратиться к данным по трудозатратам и временных затратах на разработку тест кейсов в продуктовой компании с большим web-приложением. Рассмотрим таблицы трудозатрат на создание тестовой документации в крупной продуктовой компании при появлении новой функциональности с учетом различных работ и типов тест кейсов.

Написание тест кейсов — это трудоемкий процесс, который занимает достаточно большое количество времени и если рассматривать ситуацию, когда время затраченное на разработку, но в данном в случае – это тестирование, которое тоже учитывается при планировании задач для релиза продукта.

Благодаря тест-кейсам легче обнаружить ошибки, которые могли возникнуть при разработке, или дефекты, которые пропустили на предыдущих итерациях тестирования. Важно поддерживать документацию в актуальном виде и аккуратно документировать все функции, потому что проверяемую функциональность будут тестировать ещё много раз.

Подробное описание тест-кейсов помогает автоматизировать ручные проверки и снижает затраты на техническое обслуживание и поддержку программного обеспечения. Это облегчает оценку тестового покрытия и позволяет точнее определить время, требуемое для тестирования.

Когда тестировщик решает писать качественные тест-кейсы и хочет увеличить свою продуктивность, есть несколько ключевых моментов, которые помогут достичь этих целей.

Это применимо к любой организации и команде разработчиков программного обеспечения. Если проблемы, выявляемые метриками, постоянно игнорируются, повышения качества приложений не происходит. Игнорирование проблем не приводит к их исчезновению. Даже лучшие метрики не улучшат качество ПО, если не предпринимать никаких действий. Измерения метрик и действия обеспечивают бизнес-ценность для организации и команды в целом.

Во-первых, нужно подготовить себя профессионально и психологически для некоторых ключевых моментов, необходимых для каждого успешного тестировщика ПО в IT-индустрии. Они будут рассматриваться как “исходные данные” для работы тестировщика перед началом написания тест-кейсов. Далее необходимо определить метрики качества, задействованные в проекте,

которые используются в качестве инструмента для оценки работы тестировщика на различных этапах жизненного цикла тестирования. Это будет рассматриваться как “Выходные данные” работы тестировщика после завершения написания тест-кейсов.

Таблица 5 - Среднее время на разработку тест кейсов до внедрения

Этап	**Тип тест-кейса**	**Тест кейсы (штук)**	**Время на один тест-кейс (часы)**	**Общее время (часы)**
Анализ требований	Все	100	0.5	50
Разработка тест-кейсов	Функциональные	40	0.5	20
	Нефункциональные	20	0.5	10
	Интеграционные	15	0.5	7.5
	Регрессионные	25	0.5	12.5
Утверждение тест-кейсов	Все	100	0.1	10
Итого				110

После внедрения модели машинного обучения с помощью, которой генерация шагов в тест кейсах появляется автоматически, время, затрачиваемое на создание кейсов, сократилось. В таблице 5 видно, что затраченное время на написание тест кейсов – это большой ресурс и процент работы в тестировании, так как написание документации является неотъемлемы процессом в данном секторе информационных технологий. При

помощи внедрения автоматизации в процесс написание кейсов, время для их создания существенно сокращается, что дает возможность улучшить качество продукта. Это значительное сокращение времени можно объяснить автоматизацией процесса генерации шагов в тест-кейсах. Модель машинного обучения позволяет анализировать существующие тест-кейсы и автоматически создавать шаги, что сокращает необходимость в ручном вводе информации и ускоряет весь процесс написания тестов.

Таблица 6 - Среднее время на разработку тест кейсов после внедрения алгоритма

Этап	**Тип тест-кейса**	**Тест кейсы (штук)**	**Время на один тест-кейс (часы)**	**Общее время (часы)**
Анализ требований	Все	100	0.5	50
Подготовка данных	Функциональные	40	-	2
Разработка тест-кейсов	Нефункциональные	20	-	0.5
	Интеграционные	15	-	0.5
	Регрессионные	25	-	0.5
	Все	100	-	0.5
Утверждение тест-кейсов	Все	Все	0.1	10
Итого				62

Для работы с автоматической генерации тест кейс необходимо создать базу данных с обучающими данными (прошлыми кейсами) и обучить модель в среднем это занимает около 2-х часов, поэтому была добавлена новая строчка в таблицу для более точных результатов. Как видим по таблице время на создание кейсов существенно сократилось, общее количество часов уменьшилось с 110 до 62 часов, что является отличным результатом.

В качестве еще одного примера можно привести блок-схему, описывающий процесс влияния автоматической генерации тест кейсов на покрытие и количество найденных багов.

Рисунок 14- Схема с процессами внедрения и результатами

Автоматизация генерации тест-кейсов с помощью машинного обучения оказала значительное положительное влияние на процессы тестирования в компании. Эти улучшения позволили не только повысить качество и покрытие тестирования, но и сократить затраты и время на разработку. В результате

компания смогла улучшить качество продукта.

Третья глава посвящена разработке системы, которая выполняет задачу автоматизации тестирования web-приложения.

В рамках реализации были подготовлены входные данные, которые были составлены вручную в ходе написания тестовой документации на реально существующем крупном проекте.

С помощью данных было произведено обучение модели, проведены и получены результаты – автоматизированное генерация тест кейсов, благодаря которым уменьшилось время на написание тестовых сценариев и тем самым существенно сократилось время для написания документации на тестирование web-приложений. Завершающим этапом стала оценка влияния применения данного метода автоматизации тестирования с помощью искусственного интеллекта на существующем проекте.

Система в своем базовом функционале полностью работоспособна, однако имеет потенциал дальнейшего развития, добавления нового функционала, устранения недочетов.

ЗАКЛЮЧЕНИЕ

В рамках исследования был проведен всесторонний анализ основных современных методов тестирования web-приложений, которые отражают развитие этого направления. Проведено тщательное исследование деталей тестирования web-приложений и отмечено их отличительные особенности, такие как разноплановые направления автоматизации тестирования и адаптации к изменяющимся условиям в рамках конкретных задач. Также были изучены основные методологии оценки тестирования web-приложений, используемые компаниями. Очень важно понять, какие алгоритмы и методы используются для определения вариантов автоматизации тестирования, а также как они могут быть адаптированы к уникальным требованиям на проекте. Кроме того, был проведен анализ текущих исследования в области автоматического тестирования с использованием аналитики данных и машинного обучения.

На основе обзора литературы были описаны алгоритмы и методики обработки данных и выбора направления автоматизации тестирования. Было проведено подробное сравнение эффективности и точности нескольких подходов. По результатам исследования было выбрано направление автоматизации в виде автоматической генерации тест кейсов. Так же была выбрана модель машинного обучения LSTM, обеспечивающая максимальную эффективность и точность при обработке тестовых сценариев на входе. Для генерации тестовых сценариев был выбран алгоритм, который основан на цепях Маркова. Выбранная модель была оценена с помощьютестовых данных, подтвердивших ее высокую точность и надежность. На основе этого была создана не только модель генерации тестовых сценариев, но и методология, функциональная структура и план внедрения модели в продуктовые проекты, благодаря этому компании существенно сократят времяи финансовые ресурсы

на такой процесс, как написание тестовой документации.

Проведена оценка экономических и продуктовых характеристик автоматизированной генерации тест кейсов, которая показала, что она имеет хороший потенциал для успешного практического использования. И даже внедрена на реальном проекте в крупной продуктовойкомпании.

В ходе работы были выполнены все задачи, которые были поставлены перед началом исследования, а именно:

– проведен анализ существующих подходов к тестированию ПО;

– описана специфика автоматизации тестирования ПО для web-приложений;

– проведено теоретическое исследование ML-алгоритмов, применяемыхдля решений задач в тестировании для web-приложений;

– проведено практический анализ ML-алгоритмов;

– описан алгоритм ML для реализациираз работанной модели;

– проведён анализ построения выбранной модели для выполнения задачи с учетом всех особенностей;

– подготовлены данные для обучения модели и провести анализ;

– выполнен процесс разработки и тестирование предлагаемогорешения;

– проведена экономическая оценку продуктовой эффективности модели, основанной на автоматизации тестирования с помощью ИИ.

Разработанное приложение удовлетворяет поставленным требованиям технического задания. Автоматизация тестирования web-приложений происходит благодаря автоматической генерации тестовых сценариев.

Полученные в ходе исследования обширные результаты свидетельствуют о значимости и перспективности применения модели для генераций тестовой документации для web-приложений. Одним из главных преимуществ автоматизации является - скорость и корректность полученных

результатов.

Данная система обладает универсальной применимостью и может быть эффективно интегрирована в широкий спектр задач, связанных с тестированием приложений. Независимо от специфики и контекста задачи, система предоставляет инструменты и механизмы, которые способствуют генерации тестовых сценариев. Система поддерживает генерацию тестовых сценариев для web-приложений все зависит от данных, которые будут загружены в систему для обучения для дальнейшего обучения. Таким образом, система обеспечивает всестороннее использование для автоматизации тестирования web-приложений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Андрейчикова. М.: Финансы и статистика, 2014. С. 424.
2. Артюхова А. С. Проблемы автоматизации тестирования и подходы к их решению/Артюхова Антонина Сергеевна
3. Барахтаев М.А., К. В. Демкин, А. Ю. Харитонов. Проблемы тестирования по встроенных системах / М.А. Барахтаев К. В. Демкин, А. Ю. Харитонов
4. Барсегян, А. Искусственный интеллект в области тестирования программного обеспечения. М.: Издательство Техносфера, 2020.
5. Бек, К. А. Автоматизация тестирования программного обеспечения. СПб.: БХВ-Петербург, 2019.
6. Беляева О. С., Е. В. Маркин. Искусственный интеллект: Современный подход / О. С. Беляева, Е. В. Маркин. — СПб: Питер, 2019. — 320 с.
7. Берестневой, Е.А. Компьютерный анализ данных: Учебное пособие / Под ред. О. Г. Берестневой, Е.А.
8. Гаскарова. М.: Высшая школа, 2015. С. 431.
9. Гребенюк В. М. Оценка целесообразности внедрения автоматизированного тестирования. М.: Науковедение, 2013. 128 с.
10. Джексон, М. Введение в тестирование программного обеспечения. М.: Издательство «БХВ-Петербург», 2018. и случайные процессы: учебное пособие. — Минск: Вышэйшая школа, 2012. — 720с.
11. Иваненко Р. П. Методы искусственного интеллекта / Р. П. Иваненко. — Москва: Финансы и статистика, 2005. — 624 с.
12. Интеллектуальные информационные системы: Учебник для вузов / Под ред. А.В. Аброськиной. Саратов: СГСЭУ, 2014.
13. Интеллектуальные информационные системы: Учебник для вузов / Под ред. Д. В.
14. Кирпичников А.А., Н. В. Золотянцева, Е. Г. Деев, Д.В. Ландарев.

Алгоритмы машинного обучения и искусственный интеллект в инженерии: практические примеры применения/ А.А. Кирпичников, Н. В. Золотянцева, Е. Г. Деев, Д.В. Ландарев. — Москва: Изд-во МФТИ, 2020. — 184 с.

15. Кирьянова Л. В., Лемин А.Ю., Мацеевич Т. А. Теория случайных процессов: курс лекций. — М.: Московский государственный строительный университет, 2016. — 96

16. Коллинз, Д. Автоматизация тестирования с помощью Selenium и Python. М.: ООО «Питер», 2017.

17. Котельников Е.В., Клековкина М.В. Автоматический анализ тональности текстов на основе методов машинного обучения/ 2012. – С. 27.

18. Кравцов М. Б., В.Л. Шахненко. Методы исследования операций и искусственный интеллект: учебное пособие / М. Б. Кравцов, В.Л. Шахненко. — М.: МАКС Пресс, 2017. — 256 с.

19. Крупин А. В., А. С. Глушец.,Системы искусственного интеллекта и промышленная робототехника в строительстве / А. В. Крупин, А. С. Глушец. — М.: Вузовская книга, 2018. — 256 с.

20. Кудинов М. С. Статистическое моделирование русского языка с помощью нейронных сетей: Дис. д-ра тех. наук. М., – 2016. – 106 с

21. Лабинский Ю.А. Автоматизация процесса тестирования программного обеспечения. М.: «ЛЦ Инфра-М», 2016.

22. Лаврищева Е. М. Software engineering компьютерных систем. Наукова думка, 2013. 283 с.

23. Леонтьев Н. И., В. Г. Веремейчик. Интеллектуальные системы и технологии. Учебное пособие / Н. И. Леонтьев, В. Г. Веремейчик. — Минск: БГУИР, 2016. — 232 с.

24. Логвинова К. В. Современные технологии и средства разработки программного обеспечения/ К.В. Логвинова

25. Маталыцкий М.А., Хацкевич Г. А. Теория вероятностей, математическая статистика.

26. Мерфи С., Программирование нейронных сетей на Python / С. Мерфи. — СПб: БХВ- Петербург, 2018. — 400 с.

27. Муратовой, А. М. Уразаевой. Томск: Изд-во ТПУ, 2013. С. 204

28. Палмер, Р. А. Интеграция и тестирование программного обеспечения. М.: Издательский дом "Символ-Плюс", 2019.

29. Попкова Г. И., В. Ю. Красникова. Методы искусственного интеллекта и интеллектуальные системы управления / Под ред. Г. И. Попкова, В. Ю. Красникова. — М.: Горячая линия-Телеком, 2019. — 296 с.

30. Рубин, К. Введение в автоматизированное тестирование. М.: Издательство «Лань», 2015.

31. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет стартапах. — М.: Дело. 2007.

32. Савин, С. Автоматизация тестирования на основе искусственного интеллекта. М.: ООО «БЭК», 2020.

33. Савин, С. Методы тестирования программного обеспечения. М.: Издательство «Питер», 2018.

34. Сивов С. В.,Разработка программного обеспечения в строительстве / С. В. Сивов. — М.: МГУС, 2020. — 312 с.

35. Степанченко И. В. Методы тестирования программного обеспечения: Учеб. пособие. Волгоград: ВолгГТУ, 2006. 76 с.

36. Тамре, Л. Введение в тестирование программного обеспечения; Пер.с англ. М.: Издательский дом «Вильямс», 2003. 368 с.

37. Тарасов, В. Н. Теория вероятностей, математическая статистика и случайные процессы: учебное пособие / В. Н. Тарасов, Н. Ф. Бахарева. — Самара: Поволжский государственный университет телекоммуникаций и информатики, 2017. — 283 с.

38. Томас, Д. Практика автоматизации тестирования программного обеспечения. М.: Питер, 2017.

39. Трофимова Е. В., Туральчук К.А. Разработка рекомендательной

системы на основе анализа тональности текста // Актуальные проблемы гуманитарных и естественных наук. – 2015. – №1. – С. 93–94

40. Финск, Я. Тестирование программного обеспечения: современные методы и инструменты. М.: Издательство «Лори», 2016.

41. Хартман, Дж. Методы автоматизации тестирования программного обеспечения. М.: Издательство Техносфера, 2019.

42. Хрущева, И. В. Основы математической статистики и теории случайных процессов: учебное пособие/ И. В. Хрущева, В. И. Щербаков, Д. С. Леванова. — Санкт-Петербург: Лань, 2021. — 336 с

43. Чернышевой Г. Ю. Интеллектуальный анализ данных: Учебное пособие / Под ред. Г. Ю. Чернышевой

44. Чуракова, Н. В. Искусственный интеллект в современном мире: тенденции и перспективы развития. М.: Издательство «Экономика», 2020.

45. Шапир, Дж. Тестирование на проникновение: методы и практики. М.:Издательство Бином-Пресс, 2018.

46. Штефана Э., И. Диенси, С. Б. Костенко, А.Г. Чентсов /Глубокое обучение. Пер. с англ. Э. Штефана / И. Диенси, С. Б. Костенко, А.Г. Чентсов. — Москва: ДМК Пресс, 2019. — 288 с.

47. Янг, Э. А. Автоматизация тестирования программного обеспечения с использованием JUnit. М.: Издательство «Вильямс», 2017.

48. Ясницкого Л.Н., Введение в искусственный интеллект: Учебное пособие для студ. высш. учеб. заведений Издательский центр «Академия» Л.Н. Ясницкого. М, 2015. — С. 176.

49. Greff K., R.K. Srivastava, J. Koutnk, B.R. Steunebrink, J. Schmidhuber, IEEE Trans. Neural Netw. Learn. Syst. 28, 2222 (2017)

50. Hasan A. et al. Machine learning-based sentiment analysis for twitter accounts //Mathematical and Computational Applications. – 2018. – Т. 23.-№. 1. – P. 11.

51. Iacus S., Porro G., Salini S., and Siletti E., ‗‗An Italian composite

subjective well-being index: The voice of Twitter users from 2012 to 2017, ‘‘Social Indicators Res., vol. 149, p. 1–19, 2020.

52. Peskisheva T.A. Methods for Sentiment Analysis of Natural Language Texts. // Obshhestvo. Nauka. Innovacii (NPK-2017). 2017. P. 1730-1742.

53. Z. Zhao, A. Srivastava, L. Peng, Q. Chen, ACM J. Emerg. Technol. Comput. Syst. 15, 13 (2019)

54. Pandas: офиц. сайт. – URL: https://pandas.pydata.org/about/ (дата обращения: 10.05.2024)

55. Scikit-learn: офиц. сайт. – URL: https://scikit-learn.org/stable/ (дата обращения: 10.05.2024)

56. Retentioneering: офиц. сайт. – URL: https://retentioneering.com/ (дата обращения: 10.05.2024)

ПРИЛОЖЕНИЕ А

Id	Direction	Section	TestCaseName	Automated	Preconditions
1260	Технадзор	Страница входа	Авторизация для подрядчиков через логин и пароль	False	
					Я перехожу на страницу ау
1275	Технадзор	Страница входа	Выход из аккаунта	False	
					Выполнен вход в Технадзо
1404	Технадзор	Страница входа	Смена пароля через кнопку "Я забыл свой пароль"	False	
					Я перехожу на страницу ау
					Я имею данные тестового

Рисунок А.1 –Тест кейсы до обработки

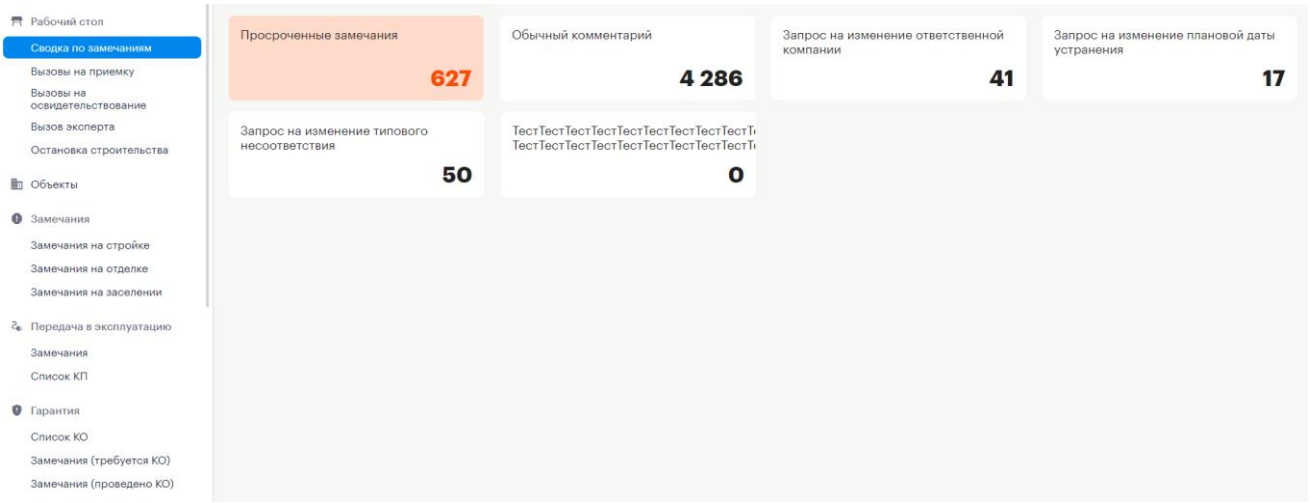


Рисунок А.2 – Интерфейс web-приложения