ISSN 2225-9309 (Print)
2025. № 1 (32). С. 124–128

Молодежный
Вестник УГАТУ

http://journal.ugatu.su

УДК 504.064.36

doi 10.54708/22259309_2025_132124

Решение задачи автоматизированного определения приоритетов при регрессионном тестировании

Э. М. Чембарисов[1], О. Н. Сметанина[2]

[1] lawluker@gmail.com [2] smoljushka@mail.ru

ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ)

Аннотация. В данной работе представлены разработка и реализация модели машинного обучения для автоматизации процесса приоритизации тестирования REST API на основе ключевых критериев, таких как количество ошибок, важность HTTP-методов, влияние на состояние базы данных, частота вызовов и количество зависимостей. Были рассмотрены различные методы классификации данных и построены модели, способные классифицировать API-методы по уровням приоритета: низкий, средний и высокий.

Ключевые слова: тестирование; REST; API.

ВВЕДЕНИЕ

С ростом объемов и сложности программного обеспечения регрессионное тестирование становится критически важным.[1] Исследования показывают, что до 40–50 % затрат на разработку уходит на тестирование и исправление ошибок. Традиционные подходы часто оказываются неэффективными из-за большого количества тестовых случаев и ограниченных ресурсов, что может привести к игнорированию важных тестов или избыточным проверкам.

С методологиями Agile и DevOps, требующими быстрой доставки обновлений, оптимизация процессов тестирования становится особенно актуальной. Многие организации по-прежнему полагаются на ручное управление тестами, не используя возможности автоматизации, что создает риски для качества продукта и репутации компании.

Разработка интеллектуальной системы для автоматического определения приоритетов тестовых случаев способна повысить эффективность регрессионного тестирования и улучшить качество программного обеспечения, что в конечном итоге приведет к повышению удовлетворенности пользователей и конкурентоспособности компании.

В статье рассматриваются вопросы постановки задачи классификации тестов с учетом факторов и приоритетов, алгоритма решения задачи с использованием методов машинного обучения и анализа данных, разработки модели и сравнения результатов прогнозов, их применения для оптимизации процессов тестирования и повышения качества программного обеспечения.

ПОСТАНОВКА ЗАДАЧИ

Задача классификации тестов для API заключается в автоматическом определении приоритета каждого теста на основе заранее заданных критериев. Приоритет теста может зависеть от различных факторов, таких как:

1. Критичность функциональности: тесты, проверяющие ключевые функции API, должны иметь более высокий приоритет.

2. Частота использования: тесты, которые относятся к часто используемым эндпоинтам, также требуют повышенного внимания.

3. История дефектов: если определенный функционал ранее вызывал проблемы, связанные с дефектами, его тесты должны быть отнесены к более высокому приоритету.

4. Изменения в коде: тесты, касающиеся недавно измененных участков кода, должны быть выполнены в первую очередь.

Перечисленные факторы определены экспертами.

Для решения задачи классификации тестов можно использовать различные методы машинного обучения и анализа данных, в основные этапы которых включают:

1. Сбор данных: необходимо собрать данные о существующих тестах, включая их описание, связанную функциональность, историю выполнения и дефектов.

2. Предобработка данных: данные необходимо подготовить к анализу путем их очистки и нормализации текстовой информации, а также кодированием категориальных переменных.

3. Выбор модели: для задачи классификации могут быть использованы различные алгоритмы, например, такие как логистическая регрессия, деревья и нейронные сети.

4. Обучение и валидация модели: модель обучается на размеченных данных, после чего проводится валидация для оценки ее эффективности.

5. Применение модели: после успешной валидации модель может быть использована для автоматической классификации новых тестов.

МЕТОДЫ КЛАССИФИКАЦИИ ДЛЯ РАССТАНОВКИ ПРИОРИТЕТОВ

Для решения задачи рассмотрим следующие методы классификации:

- Деревья решений (Decision Tree) представляют собой простой и интерпретируемый метод классификации. Они разбивают данные на подмножества на основе значений признаков, что позволяет легко визуализировать процесс принятия решений. Однако деревья решений могут быть подвержены переобучению, особенно при наличии большого количества признаков. Преимущества включают отсутствие необходимости в масштабировании данных и хорошую работу с категориальными переменными. Недостатками являются чувствительность к небольшим изменениям в данных и склонность к переобучению на небольших наборах.
- Метод случайного леса (Random Forest) – это ансамблевый метод, являющийся улучшенной версией дерева решений, что позволяет избежать проблемы с переобучением и увеличивает точность предсказания модели. Он хорошо работает с различными типами данных и обеспечивает высокую производительность. Однако его интерпретируемость ниже, чем у одиночного дерева, и он требует больше вычислительных ресурсов. Random Forest эффективно справляется с большими наборами данных, но время обучения увеличивается с количеством деревьев.
- Нейронные сети способны моделировать сложные зависимости и паттерны в данных, что делает их мощным инструментом для классификации. Они демонстрируют высокую точность при работе с большими объемами данных, однако для их обучения требуется значительное количество вычислительных ресурсов и времени. Кроме того, их интерпретируемость довольно низка, что усложняет понимание процесса принятия решений.

Эффективность нейронных сетей увеличивается с ростом объема данных.

- Метод k-ближайших соседей (KNN) является простым в реализации алгоритмом, который не требует предварительного обучения. Однако он может быть чувствителен к шуму и неэффективен с высокоразмерными данными. В процессе предсказания KNN требует значительных вычислительных ресурсов из-за необходимости вычисления расстояний до всех точек. Интерпретируемость этого метода высокая, что позволяет легко понять процесс классификации. KNN хорошо работает на небольших и средних наборах данных.
- Метод опорных векторов (SVM) демонстрирует высокую эффективность в пространствах с большим числом измерений и хорошо работает с линейно разделимыми

данными. Тем не менее он может быть чувствителен к выбору параметров и типа ядра, а также может иметь низкую скорость обработки при работе с большими объемами данных.

Умеренные требования к вычислительным ресурсам делают SVM подходящим для многих задач. Интерпретируемость метода умеренная, так как можно анализировать важность признаков, но не всегда очевидно, как происходит классификация. SVM эффективно работает на небольших и средних наборах данных.

- Логистическая регрессия – это простой и быстрый метод классификации, который хорошо работает при линейной зависимости между признаками и целевой переменной. Несмотря на свои преимущества, она ограничена линейными границами разделения и чувствительна к мультиколлинеарности. Логистическая регрессия имеет низкие требования к вычислительным ресурсам и быстро обучается. Высокая интерпретируемость позволяет легко понять влияние признаков на вероятность класса. Этот метод также хорошо подходит для небольших и средних наборов данных.

СБОР ДАННЫХ

Основные параметры для сбора данных включают в себя:

- Количество зарегистрированных ошибок для каждого API-метода, собранные из инцидентов с продуктового контура за последние 14 дней. Данный временной промежуток обусловлен длительностью спринта для команды.
- Важность HTTP-методов (GET, POST, PUT, DELETE) и их влияние на состояние базы данных (например, изменение данных или получение информации). Решение по расстановке параметров было принято экспертно, исходя из требований бизнеса и понятий безопасности и идемпотентности методов (табл. 1).

Таблица 1

Таблица безопасных и идемпотентных методов

	GET	POST	PUT	PATCH	DELETE
Безопасный	+	-	-	-	-
Идемпотентный	+	-	+	-	+

- Частота вызовов каждого метода, полученная из системы логирования, аналогична количеству зарегистрированных ошибок за тот же временной промежуток 14 дней.
- Количество зависимостей от состояний вызовов других методов, согласно бизнес-логике работы тестируемого API и наличия интеграций с внешними системами. В качестве данных для этих параметров использовались данные из системы логирования тестового раннера, в котором записывается весь путь вызовов методов API для установки необходимого для тестирования состояния и наличие в логах обращения к шлюзу внешних систем и сервисов.

Для более глубокого анализа приоритетов в тестировании следует также учитывать поддерживаемость методов на основании частоты и характера изменений в кодовой базе, хранящейся в репозитории, например, GitLab.

Частота обновлений может служить индикатором того, насколько активно используется тот или иной метод API, что делает его более критичным для стабильности системы. Методы, подвергающиеся частым изменениям, могут требовать более тщательного тестирования и мониторинга для избежания потенциальных ошибок.

Учитывая эти аспекты при оценке приоритетов, команда сможет более эффективно распределить ресурсы тестирования и сосредоточиться на наиболее уязвимых и значимых частях API.

АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

По результатам анализа проведенных экспериментов (рис. 1) видим высокое качество показателей точности методов древа решений (Decision Tree Classifier) и случайного леса (Random Forest Classifier), что хорошо соотносится с требованиями рассматриваемой задачи, благодаря высокой интерпретируемости данных методов на относительно небольшом наборе. Однако такая высокая точность может указывать на возможное переобучение моделей, что требует дальнейшего анализа.

Метод логистической регресии (Logistic Regression) показал хорошую точность в 90,91 %, что делает его надежным выбором для данной задачи, хотя и уступает по эффективности деревообразным методам. В то же время метод k-ближайших соседей (KNeighbors Classifier) и метод опорных векторов (SVC) продемонстрировали одинаковую точность 63,64 %, что может быть связано с недостаточной сложностью модели для данной задачи или необходимостью улучшения предобработки данных. В целом результаты подчеркивают важность выбора подходящей модели в зависимости от специфики задачи и качества данных.

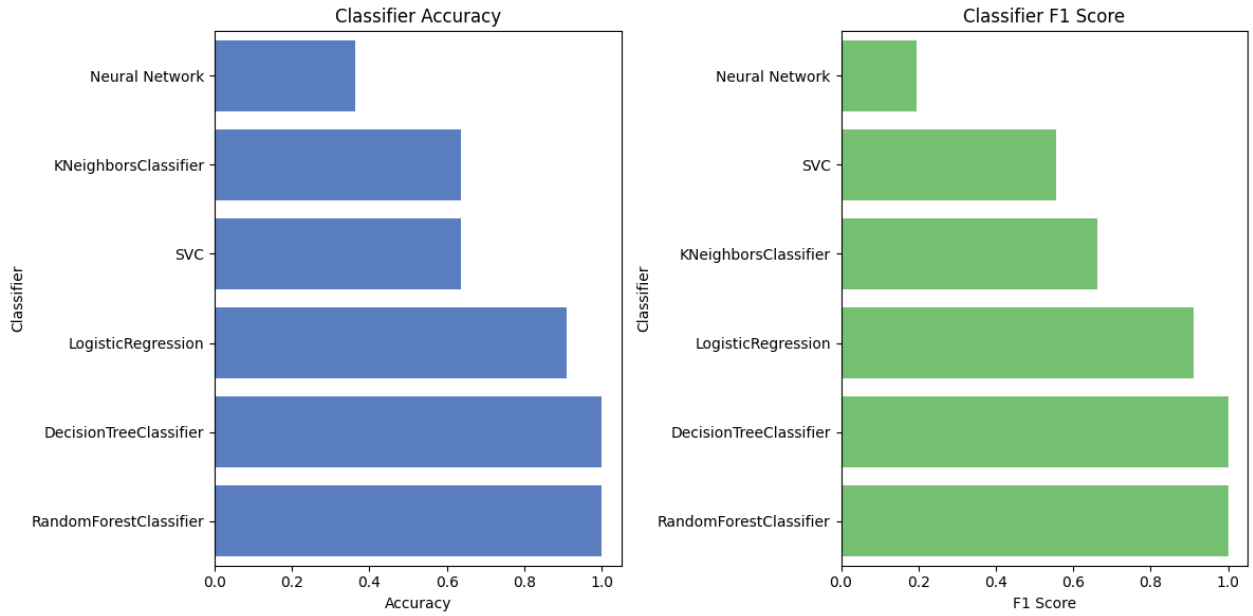

Рис. 1. Графики показателей точности предсказаний рассмотренных моделей

ЗАКЛЮЧЕНИЕ

В данной статье был проведен сравнительный анализ различных методов классификации на основе показателей Accuracy и F1-меры. В будущем планируются оптимизация рассмотренных моделей и возможности дальнейшей интеграции модели в процессы CI/CD. На текущем этапе можно однозначно сказать, что реализация моделей оценки приоритетов тестов является успешным шагом к автоматизации и оптимизации процессов тестирования. Несмотря на достигнутые результаты, всегда есть возможности для дальнейшего улучшения и адаптации модели к изменяющимся условиям и требованиям проекта.

СПИСОК ЛИТЕРАТУРЫ

1. **Данилов А. Д., Мугатина В. М.** Решение задачи оптимизации регрессионного тестирования с использованием нейросетевого подхода // Моделирование, оптимизация и информационные технологии. 2020. Т. 8. №. 1 (28). С. 35.

2. **Sugato B.** Semi-supervised Clustering: Probabilistic Models, Algorithms and Experiments http://www.cs.utexas.edu/users/sugato/papers/sugatophdthesis.pdf

3. **State** of Agile Report. https://stateofagile.com.

ОБ АВТОРАХ

Чембарисов Эмиль Марсович, студ**.** каф. ВМиК.

Сметанина Ольга Николаевна, профессор, д/н, доцент каф. ВМиК

METADATA

Title: Solving the Problem of Automated Prioritization in Regression Testing

Author: E.M. Chembarisov, O.N. Smetanina

Affiliation:

[1,2] Ufa University of Science and Technology (UUST), Russia.

Email: [1] lawluker@gmail.com [2] smoljushka@mail.ru

Language: Russian.

Source: Molodezhnyj Vestnik UGATU (scientific journal of Ufa University of Science and Technology), no. 1 (32), pp. 124-128, 2025. ISSN 2225-9309 (Print).

Abstract: This paper presents the development and implementation of a machine learning model to automate the process of prioritizing REST API testing based on key criteria such as the number of errors, the importance of HTTP methods, the impact on the database state, the frequency of calls, and the number of dependencies. Various data classification methods were considered and models were built that can classify API methods by priority levels: low, medium, and high.

Keywords: software testing, REST, API

About authors:

Chembarisov Emil Marsovich, student, Dept. of Computational Mathematics and Cybernetics (UUST).

Smetanina Olga Nikolayevna, professor, Dept. of Computational Mathematics and Cybernetics (UUST).