DOI 10.37882/2223-2966.2023.05.07

АНАЛИЗ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ С ЦЕЛЬЮ ОБНАРУЖЕНИЯ ДЕФЕКТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ANALYSIS OF MACHINE LEARNING METHODS TO DETECT SOFTWARE DEFECTS

T. Vishnevskaya
I. Klimov

Summary: This article provides an overview of software defects that occur both at the development stage and in the process of using the software product. There are three groups in the classification of methods for their detection: static, dynamic and operational. The possibility of using machine learning to detect defects in the software being developed is considered. An overview of the most common machine learning methods (naive Bayesian classifier, support vector machine, decision tree, random forest, boosting) and the results of comparing these methods based on selected metrics are presented. In conclusion, it is concluded that the gradient boosting method is the most effective and promising for the problem of detecting software defects.

Keywords: defect, machine learning, detection, methods.

Вишневская Татьяна Ивановна
к.ф-м.н., доцент, Московский государственный технический университет им. Н.Э. Баумана
iu7mag@mail.ru
Климов Илья Сергеевич
Московский государственный технический университет им. Н.Э. Баумана
ilyasssklimov@gmail.com

Аннотация. В данной статье представлен обзор дефектов программного обеспечения (далее — ПО), встречающиеся как на стадии разработки, так и в процессе использования программного продукта. Выделены три группы в классификации методов для их обнаружения: статические, динамические и эксплуатационные. Рассмотрена возможность применения машинного обучения для обнаружения дефектов разрабатываемого ПО. Представлен обзор наиболее распространенных методов машинного обучения (наивный байесовский классификатор, метод опорных векторов, дерево решений, случайный лес, бустинг) и результаты сравнения этих методов на основе выбранных метрик. В заключении сделан вывод о том, что для задачи обнаружения дефектов ПО наиболее эффективным и перспективным является метод градиентного бустинга.

Ключевые слова: дефект, машинное обучение, обнаружение, методы.

Одной из актуальных проблем разработки и внедрения программного обеспечения (ПО) является наличие дефектов, то есть ошибок в коде программы, приводящих к снижению качества продукции. Причинами появления дефектов могут стать некачественная организация процесса разработки ПО, недостаточная квалификация и опыт разработчиков, недостаточность ресурсов на разработку. Затраты на выявление и устранение дефектов могут составлять до 80 % от общей стоимости ПО [1]. При этом чем раньше будет обнаружен дефект, тем меньше ущерба будет нанесено разработчику и эксплуатанту ПО.

Существует множество техник и технологий для определения дефектов: начиная от ручных средств, заканчивая автоматизированным тестированием ПО. Однако, чем больше объем исходного кода, тем больше трудозатрат необходимо для поддержания качества программной системы, при этом ресурсов может не хватать. В данном случае решением могут стать методы машинного обучения, которые на основе ранее написанного кода позволят дать вероятностную оценку нахождения дефекта в том или ином месте программы.

Согласно стандартному глоссарию терминов и определений, используемых в тестировании ПО, дефект — это изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию [2]. Другими словами, дефект — это отклонение от первоначальных бизнес-требований, логическая ошибка в исходном коде программы. Дефект не оказывает влияния на функционирование ПО до тех пор, пока он не будет обнаружен при эксплуатации программы. Это может привести к тому, что продукт не будет удовлетворять потребностям пользователя, а также к отказам компонента или системы. Последствия программной ошибки для пользователя могут быть серьезны. Например, дефект может поставить под угрозу бизнес-репутацию, государственную безопасность, бизнес или безопасность пользователей, окружающую среду [3].

Дефекты в программе могут быть обнаружены не сразу и при этом иметь отрицательное влияние на процесс ее использования. При позднем обнаружении дефектов снижаются качество и надежность ПО, увеличиваются затраты на его переработку, проявляются негативные последствия. Своевременное выявление и исправление дефектов играют большую роль в жизненном цикле ПО, помогает улучшить качество разрабатываемых систем.

Для выявления дефектов ПО используются различные методы, которые можно разделить на три категории.

1. Статические методы — методы, при которых ПО тестируется без какого-либо выполнения программы (системы).
2. Динамические методы — в данных методах ПО тестируется путем выполнения программы.

3. Эксплуатационные методы — методы, при которых дефект обнаруживается пользователями, клиентами или контролирующим персоналом, то есть в результате сбоя.

Все методы важны и необходимы в процессе управления дефектам. При их объединении достигается наиболее качественный результат. На ранней стадии наиболее эффективными методами являются статические. Они позволяют снизить затраты, необходимые для исправления дефектов, и сводят к минимуму их влияние.

Машинное обучение — это математическая дисциплина, в основе которой лежат теория вероятностей, математическая статистика, численные методы оптимизации, дискретный анализ, основной целью которого является выделение знаний из имеющихся данных. Под знанием подразумевается обученная модель, способная совершать предсказания на новых поступающих данных [4]. Важность машинного обучения возрастает не только в исследованиях, имеющих отношение к компьютерным наукам, но и в нашей повседневной жизни.

Прогнозирование дефектов является важной частью цикла разработки ПО. Знание о компонентах, содержащих наибольшее число дефектов, позволяет распределить ресурсы тестирования так, чтобы в первую очередь проверялись компоненты с высокой вероятностью наличия дефектов [5]. Сложно составить правила при поиске дефектов, так как могут встретиться совершенно разные ошибки, поэтому на помощь приходят методы машинного обучения, которые решают данную проблему, обучаясь на примерах.

Исследователи применяли различные алгоритмы для решения рассматриваемой задачи. На рисунке 1 представлена общая схема процесса обучения модели обнаружения дефектов ПО.

Первым шагом построения модели является создание и идентификация положительных и отрицательных образцов из набора данных. Каждый образец может представлять собой программный компонент, файл исходного кода, класс или функцию в зависимости от выбранной степени детализации. Экземпляр имеет метрики и метки, которые указывают, склонен он к дефектам или нет. Затем они передаются в модель машинного обучения для обучения. Наконец, обученная модель может классифицировать различные фрагменты кода как ошибочные или безопасные на основе закодированных знаний [6].

Для обучения моделей во многих статьях, например, [7-12], используются классические методы машинного обучения такие, как дерево решений, случайный лес, градиентный бустинг, метод опорных векторов, наивный байесовский классификатор [6], решающие задачу классификации.

1. *Наивный байесовский классификатор*. Это метод, в котором используется простой классификатор, основанный на применении теоремы Байеса с наивным предположением о независимости.
2. *Метод опорных векторов*. Рассматривается задача классификации на два непересекающихся класса, в котором объекты описываются n-мерными вещественными векторами. Необходимо найти и построить гиперплоскость вида $w^Tx + b = 0$, разделяющую объекты на два подмножества с максимальной граничной областью.
3. *Дерево решений*. Этот метод зачастую применяется в задачах классификации. Деревья состоят из вершин, в которых записываются проверяемые условия (признаки), и листьев, в которых записаны ответы дерева (один из классов). Обучение состоит в настройке условий в узлах дерева и ответов в его листьях с целью достижения максимального качества классификации. К основным преимуществам данного метода можно отнести интерпретируемость, автоматический отбор признаков

Рис. 1. Процесс обучения модели обнаружения дефектов ПО

и управляемость. К недостаткам — зависимость от сбалансированности обучающих примеров, часто возникающее переобучение, экспоненциальное уменьшение обучающей выборки.

4. *Случайный лес*. Проблема переобучения, возникающая при использовании дерева решений, может быть решена лесом решений — обучается несколько деревьев, при этом результат определяется путем голосования. Для достижения независимости ошибок деревьев, составляющих лес решений, применяются специальные методы, например, случайный лес. Для каждого дерева случайным образом выбираются объекты из обучающей выборки. При этом некоторые могут быть выбраны несколько раз, а некоторые вовсе пропущены. В результате создается новое подмножество, на котором и обучается модель.
5. *Бустинг*. Это модификация алгоритма случайного леса, обучение происходит путем последовательного обучения нескольких моделей для повышения точности всей системы. Выходным данным отдельных деревьев присваиваются веса. Затем неправильным классификациям из первого дерева решений присваивается больший вес, после чего данные передаются в следующее дерево. После многочисленных циклов бустинг объединяет «слабые» классификаторы в одну модель. Существуют две основные разновидности бустинга: адаптивный и градиентный. Разница состоит в том, что градиентный бустинг не присваивает неправильно классифицированным элементам больший вес. Вместо этого модель оптимизирует функцию потерь, используя градиентный спуск, в результате чего текущая базовая модель всегда становится эффективнее предыдущей [13].

Для задачи обнаружения дефектов ПО в результате классификации объекты помечаются как положительные (имеются дефекты) и отрицательные (дефекты отсутствуют). Для оценки качества работы полученных моделей используются различные метрики. Так, в статьях [8–12] при сравнении результатов рассматриваются различные метрики.

1. Accuracy (точность) — широко используемая метрика, представляет собой отношение всех правильных прогнозов к общему числу предсказанных образцов.
2. Precision (точность) — это доля прогнозируемых положительных результатов, которые действительно относятся к этому классу, от всех положительно предсказанных объектов.
3. Recall (полнота) — пропорция всех верно-положительных предсказанных объектов к общему количеству действительно положительных. Чем выше значение полноты, тем меньше положительных примеров пропущено в классификации.
4. F-measure (F-мера) — взвешенное гармоническое среднее полноты и точности. Этот показатель демонстрирует, как много объектов классифицируется моделью правильно, и сколько истинных экземпляров она не пропустит [14].

Для сравнения рассмотренных методов использованы результаты, описанные в статьях [8-12]. Модели обучались на данных, представленных в репозитории PROMISE [15]. Данные представляют собой набор метрик Маккейба и Холстеда для модулей, написанных на языках программирования C и C++. В таблице 1 отображены средние арифметические значения для каждой из метрик сравнения по всем наборам данных и рассматриваемым статьям.

Таблица 1.

Сравнительная таблица результатов работы методов

Метрики / Методы	Accuracy (точность)	Precision (точность)	Recall (полнота)	F-measure (F-мера)
Наивный байесовский классификатор	0.795	0.845	0.803	0.849
Метод опорных векторов	0.841	0.901	0.879	0.902
Дерево решений	0.823	0.845	0.878	0.889
Случайный лес	0.845	0.859	0.863	0.890
Градиентный бустинг	0.847	0.903	0.883	0.903
Адаптивный бустинг	0.835	0.858	0.861	0.889

Таким образом, по каждой из метрик метод градиентного бустинга показал наивысший результат. Можно сделать вывод, что его применение является наиболее выгодным для рассматриваемой задачи. Благодаря использованию ансамблю моделей и большому варьированию параметров при обучении, данный метод имеет высокую перспективу использования.

В данной работе был представлен обзор дефектов разрабатываемого ПО, выделены три основные группы методов для их обнаружения: статические, динамически и эксплуатационные. Рассмотрена возможность использования машинного обучения для решения задачи обнаружения дефектов ПО. Описаны наиболее распространенные алгоритмы, применяемые этой цели: наивный байесовский классификатор, метод опорных векторов, дерево решений, случайный лес, бустинг. Также описаны метрики для сравнения качества моделей и проведено сравнение методов машинного обучения для обнаружения дефектов ПО. На основе этого выбран наиболее эффективный метод — статический анализ кода для поиска программных дефектов с использованием алгоритма градиентного бустинга.

ЛИТЕРАТУРА

1. Викторов Д.С., Жидков Е.Н., Жидков Р.Н. Методика статического анализа для поиска дефектов естественной семантики программных объектов и ее программная реализация на базе инфраструктуры компилятора LLVM и фронтенда Clang // Журнал Сибирского федерального университета. 2018. — С. 801–810.
2. International Software Testing Qualifications Board. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения. Версия 2.3 (от 9 июля 2014 года): ред. пер. Александр Александров. С. 17.
3. ГОСТ Р 56920 — 2016. Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения. М.: Стандартинформ, 2016. С. 9.
4. Платонов А.В. Машинное обучение: учебное пособие для вузов Москва: Издательство Юрайт, 2022. С. 85. ISBN 978-5-534-15561-7.
5. Юхименко Н.В., Белов Ю.С. Обзор методов прогнозирования дефектов программного обеспечения // Программные продукты, системы и алгоритмы. 2019. №1. С. 2.
6. Sharma T., Kechagia M., Georgiou S., Tiwari R., Vats I., Moazen H., Sarro F. A Survey on Machine Learning Techniques for Source Code Analysis. 2022. P. 11–13.
7. Assim A., Obeidat Q., Hammad M. Software Defects Prediction using Machine Learning Algorithms // 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI). 2020. P. 1–6.
8. Shah M., Pujara N. Software Defects Prediction Using Machine Learning. 2020. P. 1–5.
9. Iqbal A. Aftab S., Ali U., Husen A. Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets // International Journal of Advanced Computer Science and Applications. 2019. P. 1–6.
10. Aleem S., Capretz L.F., Ahmed F. Comparative performance analysis of machine learning techniques for software bug detection. 2015. P. 1–9.
11. Cetiner M., Sahingoz O.K. A Comparative Analysis for Machine Learning based Software Defect Prediction Systems // 11th International Conference on Computing, Communication and Networking Technologies. 2020. P. 1–7.
12. Bhandari G.P., Gupta R. Machine learning based software fault prediction utilizing source code metrics // IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu (Nepal). 2018. P. 1–6.
13. Кафтанников И.Л., А.В. Парасич А.В. Особенности применения деревьев решений в задачах классификации // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». 2015. С. 26–32.
14. Дудченко П.В. Метрики оценки классификаторов в задачах медицинской диагностики // Молодежь и современные информационные технологии: сборник трудов XVI Международной научно-практической конференции студентов, аспирантов и молодых ученых. 2019. С. 164–165.
15. NASA. Promise software engineering repository. URL: http://promise.site.uottawa.ca/SERepository/datasets-page.html (Дата обращения: 29.11.2022).

Журнал «Современная наука: актуальные проблемы теории и практики»