Объектно-ориентированное программирование — это парадигма, используемая значительным числом разработчиков при проектировании, разработке и внедрении программных систем. Она облегчает этап сопровождения разработанных приложений. К сожалению, этот этап все чаще подвергается опасности из-за того, что разработчики вносят дефекты в разработку, которые негативно влияют на качество программного обеспечения. Плохое обслуживание препятствует развитию системы, простоте внесения изменений, которые могут внести инженеры-программисты, пониманию программы и увеличивает склонность к ошибкам. В итоге, плохое обслуживание приводит к ухудшению качества программного обеспечения и сокращению срока службы систем. Эти аномалии, которые не являются ошибками или технически некорректными кодами и не нарушают работу программы немедленно, указывают на слабые места в дизайне и могут замедлить разработку или увеличить риск ошибок или сбоев в будущем. Они могут быть идентифицированы на основе таксономии подходов к обнаружению, представленной Hadj-Kacem *et al*., из четырех источников информации: структурного, семантического, поведенческого и исторического. Структурные дефекты, по сути, относятся к структуре кода, которая нарушает принципы объектно-ориентированного проектирования, такие как модульность, инкапсуляция, абстракция данных и т. д. Blob и Long Method — это два структурных дефекта, которые согласуются с этим утверждением и широко распространены в исходном коде.

Они классифицируются в соответствии с несколькими абстракциями, которые мы группируем в две категории: традиционные эвристические подходы и подходы, основанные на машинном обучении. Из-за сложностей с поиском пороговых значений для идентификации метрик, отсутствия согласованности между различными методами идентификации, субъективности разработчиков в определении дефектов и сложности ручного построения оптимальных эвристик исследования сместились в сторону машинного обучения. Эти модели представляют собой математические методы, которые используют исторические данные для автоматического выявления сложных закономерностей и принятия обоснованных и разумных решений. Тем не менее, эта новая парадигма в основном строится на моделях обучения, взятых по отдельности, оставляя в стороне поговорку о том, что сила в цифрах, и редко принимает во внимание дисбаланс данных в контексте исходных кодов.

Зная, что предварительно обученные модели обладают способностью извлекать нюансы и контекстную информацию, они могут обеспечить лучшее различение между классами меньшинства и большинства в контексте несбалансированных наборов данных.

Какой метод машинного обучения подходит для несбалансированных наборов данных в контексте обнаружения дефектов структурного развития?

Таким образом, наша главная цель состоит в том, чтобы построить модель, основанную на ансамблевом обучении, базовый оценщик которой состоит из предварительно обученной модели, способной внести свой вклад в баланс классов, и классификатора.

Исходя из вышеизложенной цели, возникают основополагающие вопросы.

RQ1: Каково оптимальное количество базовых оценок для модели?

RQ2: Смягчают ли предварительно обученные модели классовый дисбаланс?

ВОПРОС 3: Является ли метод ансамблевого обучения, использующий предварительно обученную модель для векторных представлений, включающий классификатор глубокого обучения, самым современным для ансамблевых методов?

Чтобы ответить на эти вопросы, мы организуем эту статью следующим образом: Во втором разделе мы представляем обзор литературы по подходам к обнаружению и методам ансамблевого обучения. В разделе 3 представлен наш подход. В разделе 4 мы представляем и обсуждаем наши результаты, и, наконец, завершаем статью.

**2. Сопутствующие работы**

В области подходов к выявлению дефектов структурного развития определено несколько классификаций. В данной работе мы суммируем классификацию на две группы, как указали Yue *et al:* традиционные эвристические подходы и подходы, основанные на машинном обучении.

**2.1. Традиционные эвристические подходы**

Процесс эвристических подходов, как правило, разворачивается в два этапа. Вычисляется набор метрик, связанных или не связанных с конкретными показателями на экземплярах кода, характеризующих рассматриваемый дефект. Затем к этим метрикам применяются пороговые значения. Следуя этой схеме, Пельдсус *и др*. предложили модель, которая связывает метрики программного обеспечения и различные индикаторы запахов кода, чтобы позволить системе не ухудшаться по мере ее развития. Однако субъективность индикаторов запаха кода может сделать инструменты обнаружения непригодными для использования в определенных контекстах.

Чен *и его коллеги* одновременно внедрили инструмент Pysmell, стратегия обнаружения которого включала применение набора метрик, связанных с параметризованными пороговыми значениями, к соответствующим фрагментам кода. Аналогичным образом, Хаммад *и др*. разработали плагин под названием JFly (Java Fly), интегрированный в среду Eclipse на основе набора правил, характеризующих дефекты, которые необходимо обнаружить, включая программные метрики, связанные с пороговыми значениями. Все эти подходы используют пороговые значения, ставя повторяющуюся проблему субъективности оптимального выбора.

В методах обнаружения дефектов все чаще отказываются от традиционных эвристических подходов, в том числе из-за субъективности в определении пороговых значений, направляя исследования в сторону машинного обучения.

**2.2. Подходы, основанные на машинном обучении**

Машинное обучение — это область исследований в области искусственного интеллекта, которая направлена на то, чтобы дать машинам возможность «учиться» на данных с помощью математических моделей. Два метода использования алгоритмов машинного обучения представляют собой современное состояние исследований.

**2.2.1. Отдельные модельные случаи**

Hamdy *et al*. экспериментировали с двумя рекуррентными нейронными сетями, LSTM (Long Short Term Memory) и GRU (Gated recurrent unit), а также сверточной нейронной сетью CNN для обнаружения блобов. Они пришли к выводу, что нейронные сети превосходят широко используемые модели машинного обучения, такие как наивный алгоритм Байеса, случайные леса и деревья принятия решений.

Несмотря на то, что добавление лексических и синтаксических особенностей исходного кода в метрики программного обеспечения обеспечило набор релевантной информации для обучающих данных, эти функции не учитывают семантику кода.

Kacem *et al*. провели исследование с использованием гибридного метода, сочетающего фазу обучения без учителя с использованием глубокого автоэнкодера, целью которого является преобразование фрагментов кода в векторные представления уменьшенных размерностей и контролируемое обучение (искусственная нейронная сеть) для классификации этих кодов на основе их векторных характеристик.

Шарма *и др.* сравнили три типа моделей: сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN) и автоэнкодеры со скрытыми слоями, состоящими из плотных нейронных сетей (DNN), CNN и RNN. Авторы использовали инструмент Tokenizer с открытым исходным кодом для генерации векторных представлений кода.

В этих двух вышеупомянутых работах процесс векторного представления фрагментов кода не учитывает контекст кода, определяющий его смысл.

Чтобы учесть семантику исходного кода, Kacem *et al*. разработали подход, который генерирует векторные представления из абстрактных синтаксических деревьев из фрагментов кода, используемых в качестве входных параметров для вариационного автоэнкодера (VAE), который производит семантическую информацию в процессе обучения Наконец, к этой семантической информации применяется классификатор логистической регрессии, чтобы определить, является ли фрагмент кода дефектом или нет. Ограниченность этой работы заключается в процедуре извлечения репрезентативных векторов из фрагментов кода, которая включает в себя несколько шагов, потенциально делающих всю систему более сложной.