**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ*

## *НА ТЕМУ:*

## «Методы машинного обучения для определения дефектов программного обеспечения»

Студент ИУ7-52Б _____ И. С. Климов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ Т. И. Вишневская
(Подпись, дата) (И.О.Фамилия)

*2021 г.*

**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-7
_____ И. В. Рудаков
« _____ » _____ 20 ____ г.

# З А Д А Н И Е
## на выполнение научно-исследовательской работы

по теме Методы машинного обучения для определения дефектов программного обеспечения

Студент группы ИУ7-52Б

Климов Илья Сергеевич
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

учебная

Источник тематики (кафедра, предприятие, НИР) НИР

График выполнения НИР: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

***Техническое задание*** Изучить существующие решения для определения дефектов программного обеспечения. Привести сравнительную характеристику методов машинного обучения для обнаружения дефектов ПО. Сделать вывод об эффективности приведенных методов.

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 15-25 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
Презентация на 8-10 слайдах.

Дата выдачи задания « ___ » _____ 20__ г.

**Руководитель НИР** _____ Т.И. Вишневская
(Подпись, дата) (И.О.Фамилия)

**Студент** _____ И.С. Климов
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

# Содержание

# ВВЕДЕНИЕ

Одной из актуальных проблем разработки и внедрения программного обеспечения (далее – ПО) является наличие дефектов, то есть ошибок в коде программы, приводящих к негативным последствиям. Причинами появления дефектов могут стать некачественная организация процесса разработки ПО, недостаточная квалификация и опыт разработчиков, недостаточность ресурсов на разработку ПО. При этом чем раньше будет обнаружен дефект, тем меньше ущерба будет нанесено разработчику и эксплуатанту ПО.

Для решения таких задач наиболее рационально применять методы машинного обучения с созданием соответствующих моделей. Модель прогнозирования может предсказать наличие ошибки в новом экземпляре. Прогнозирование того, содержит ли данный экземпляр дефекты, представляет собой классификацию, а прогнозирование количества ошибок в экземпляре – регрессию.

**Цель данной работы**: выявить наиболее эффективные алгоритмы машинного обучения для определения дефектов ПО. Для достижения цели необходимо решить следующие **задачи**:

- изучить проблему обнаружения дефектов ПО;
- рассмотреть процесс прогнозирования дефектов ПО на основе машинного обучения;
- составить классификацию существующих методов;
- рассмотреть и проанализировать методы и модели машинного обучения на основе их классификации;
- привести сравнительную таблицу для рассмотренных методов.

# 1 Анализ предметной области

## 1.1 Определение понятия дефекта программного обеспечения

Согласно стандартному глоссарию терминов и определений, используемых в тестировании программного обеспечения, недочет или дефект (fault) – это изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию, например, неверный оператор или определение данных. Дефект, обнаруженный во время выполнения, может привести к отказам компонента или системы [1]. Другими словами, дефект – это логическая ошибка в исходном коде программы, наличие которой при определенных условиях может привести к отказу этой программы. Выделяют три вида дефектов: программные, технические, архитектурные. Программные ошибки возникают из-за несовершенства исходного кода конечного продукта. Технические дефекты сводятся к доступу тех или иных функций готового решения или его дизайна. Архитектурные ошибки – это ошибки, вызванные внешними факторами, которые заранее не были учтены при проектировании решения, вследствие чего приложение демонстрирует результат работы, отличный от ожидаемого.

Основными причинами возникновения дефектов являются сложность реализации задачи, сжатые сроки разработки, несовершенство документации, изменение требований, недостаточная квалификация и опыт разработчиков, неправильная организация процесса разработки.

## 1.2 Обнаружение дефектов ПО

Дефекты в программе могут быть обнаружены не сразу и при этом иметь отрицательное влияние на процесс ее эксплуатации. Чем дольше будут иметься необнаруженные дефекты, тем более серьезные негативные последствия будут

проявляться. При позднем обнаружении дефектов снижаются качество и надежность ПО, увеличиваются затраты на его переработку.

Поэтому задача своевременного, раннего предсказания дефектов ПО является весьма актуальной.

В настоящее время программные системы становятся все более сложными и многокомпонентными. Чем больше объем исходного кода, тем больше трудозатрат необходимо для поддержания качества программной системы на должном уровне. Своевременное выявление и исправление дефектов играют большую роль в жизненном цикле ПО. Прогнозирование того, содержит ли программный компонент дефекты проектирования, помогает улучшить качество разрабатываемых систем.

Методы прогнозирования (предсказания) дефектов могут дать ответ на следующие вопросы: каково количество необнаруженных дефектов в ПО и в каких программных компонентах они содержатся. Знание о компонентах, содержащих наибольшее число дефектов, позволяет распределить ресурсы тестирования так, чтобы в первую очередь наиболее тщательно проверялись компоненты с высокой вероятностью наличия дефектов.

К наиболее эффективным методам предсказания дефектов ПО относятся методы машинного обучения. Сущность машинного обучения состоит в использовании нужных признаков для построения моделей, подходящих для решения правильно поставленных задач. Таким образом признаки определяют «язык», на котором описываются объекты предметной области. При этом имея представления в виде признаков, можно уже не возвращаться к самим объектам предметной области. К задачам машинного обучения относятся определение прогноза или вывода, основанного на возникшей проблеме или на вопросе, а также доступных данных. Задачи машинного обучения полагаются на шаблоны в данных, а не на явное программирование [2].

# 2 Классификация существующих решений

## 2.1 Классификация методов определения дефектов ПО

В статье Н.В. Юхименко и Ю.С. Белова [3] рассматриваются методы прогнозирования дефектов ПО, которые сгруппированы в зависимости от цели – прогнозирование количества дефектов или классифицирование дефектов. На основе этих методов анализируются различные модели и алгоритмы, такие как модель роста надежности, метод исторических аналогий, конструктивная модель качества, регрессионные модели.

Также для каждой группы рассмотрены методы машинного обучения на основе статей [4] и [5]: наивный байесовский классификатор, алгоритм случайного леса, классификатор градиентного бустинга, метод опорных векторов.

На рисунке 1 приведены наиболее распространенные методы и модели машинного обучения для определения дефектов ПО, сгруппированные в зависимости от цели – прогнозирование количества дефектов или их классификация.

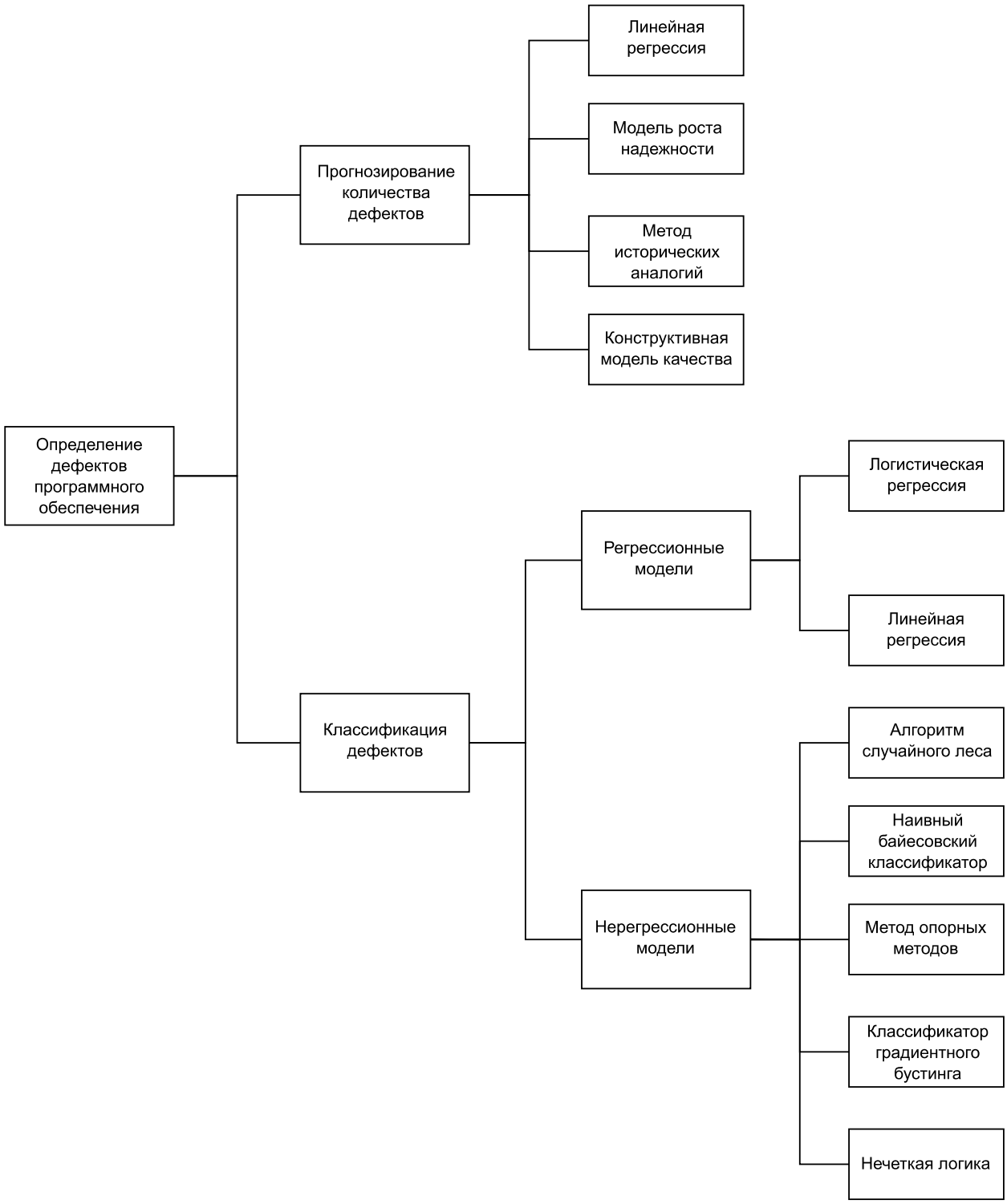


Рисунок 1 – Классификация методов и моделей определения программных дефектов

Методы машинного обучения − динамические алгоритмы обучения. Их производительность, как правило, улучшается по мере поступления дополнительных данных. На рисунке 2 показан общий процесс прогнозирования дефектов ПО на основе моделей машинного обучения.

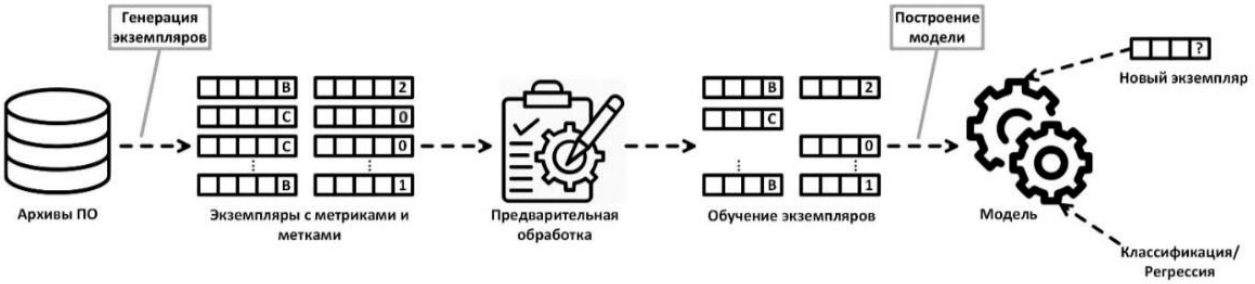


Рисунок 2 – Процесс прогнозирования дефектов ПО на основе методов машинного обучения.

Первым шагом для построения модели прогнозирования является создание экземпляров из архивов ПО (системы контроля версий, системы отслеживания ошибок, архивы электронной почты и т.д.). В свою очередь каждый экземпляр представляет собой программный компонент, файл исходного кода, класс или функцию в зависимости от выбранной степени детализации прогнозирования.

Каждый извлеченный экземпляр имеет метки, которые указывают, склонен ли данный экземпляр к дефектам (если да, то дополнительно указывается их количество). Например, на рисунке 2 экземпляры, содержащие дефекты, помечены как «B» с указанием количества ошибок, а экземпляры без ошибок − как «C».

Далее производится предварительная обработка экземпляров, включающая нормализацию данных и шумоподавление. Данный этап не является обязательным, однако его выполнение позволит получить более достоверные результаты.

После получения окончательного набора обучающих элементов можно переходить к обучению модели прогнозирования. Модель прогнозирования может предсказать наличие ошибки в новом экземпляре. Прогнозирование наличия дефекта в данном экземпляре представляет собой классификацию, а количества ошибок в нем – регрессию.

## 2.2 Методы прогнозирования количества дефектов ПО

Информация о том, сколько дефектов содержится в рассматриваемом объекте позволяет грамотно распределить ресурсы тестирования, определить наиболее уязвимое место в проекте. Подразделяют несколько видов методов: методы прогнозирования, которые оперируют только количеством дефектов, обнаруженных во время разработки и тестирования без учета других атрибутов, связанных с внутренней структурой, дизайном или реализацией продукта, называют методами черного ящика. Методы, которые используют атрибуты, связанные с процессом и продуктом, например, размер, сложность, изменения, относятся к методам белого ящика [6]. Рассмотрим наиболее распространенные модели, применяемые для прогнозирования количества дефектов ПО.

Регрессионные модели используются для решения задач, требующих изучения отношения между двумя и более переменными, позволяют на основе входных данных (в данном случае – некоторых атрибутов программы) получить значение предсказываемой переменной (количества дефектов ПО) [5]. Одной из самых распространенных регрессий является линейная. **Линейная регрессия** является одной из самых простых. Например, уравнение (1) содержит две независимые переменные X1 и X2, каждая из которых содержит информацию об искомой переменной Y.

$$Y = B_0 + B_1 X_1 + B_2 X_2 + \varepsilon, \quad (1)$$

где $B_0$, $B_1$, $B_2$ – коэффициенты регрессии;

$\varepsilon$ – стандартная ошибка модели.

Можно заметить, что связь между переменными является линейной. Задача модели – подобрать такие коэффициенты, чтобы минимизировать функцию потерь (функцию, характеризующую отклонение прогнозируемого значения от ожидаемого). Стандартная ошибка показывает, на сколько в среднем происходит ошибка. Модель обладает достаточно высокой скоростью, проста в обучении и

является легко интерпретируемой. Главным недостатком линейной регрессии является требование равномерного распределения данных.

**Модели роста надежности ПО** основаны на использовании функции надежности. Предполагается, что в процессе надежность ПО увеличивается. Надежность ПО – это вероятность того, что в определенной среде и в определенное время оно будет работать должным образом. Для получения этой функции чаще всего выбирается экспоненциальное распределение, и на основе предыдущего опыта и текущей информации об обнаруженных дефектах оцениваются его параметры [7].

**Метод исторических аналогий** основан на сборе и сравнении различных метрик между прошлым и текущим проектами. На основе установления сходства метрик по некоторым признакам появляется возможность сделать вывод о сходстве в других признаках. Для прогнозирования дефектов ПО обычно используются размер, тип приложения, функциональная сложность и другие параметры. Анализ может быть выполнен на уровне проекта, подсистемы или компонента. Модели, основанные на методе исторических аналогий, особенно полезны, когда для предметной области трудно выявить конкретные правила.

**Конструктивная модель качества** использует экспертно-детерминированные подмодели внедрения и устранения дефектов для построения качественной модели. В рамках этой модели с помощью подмодели «внедрение дефектов» (defect introduction, DI) сначала оценивается количество нетривиальных требований, вводимых дефектов проектирования и кодирования. Эта подмодель использует оценки размера ПО и прочие атрибуты, связанные с проектом и процессом (платформа и т.д.). Выходные данные DI подмодели являются входными данными для подмодели устранения дефектов (defect removal, DR). Результат подмодели DR – оценка количества оставшихся дефектов на единицу размера.

## 2.3 Методы классификации дефектов ПО

Методы классификации позволяют определить подверженные дефектам программные модули и обычно применяются на более низких уровнях детализации, например, на уровне файлов и классов.

**Логистическая регрессия** – это тип регрессионного анализа, используемый для прогнозирования результата категориальной переменной (то есть зависимой переменной, которая может принимать ограниченное число значений, величины которых не являются значимыми, но порядок этих величин может быть или не быть значимым) [8]. С помощью логистической регрессии по формуле (2) можно спрогнозировать вероятность принадлежности некоторому классу в зависимости от множества признаков.

$$\mathrm{P}(Y) = \frac{1}{1+e^{-(\theta_1 x_1 + \cdots + \theta_n x_{n)}}}, \quad (2)$$

где $\Theta_i$ – коэффициенты регрессии;

$x_i$ – независимые переменные.

Обычно для подбора параметров используется метод максимального правдоподобия, согласно которому $\theta_1 \ldots \theta_n$ выбираются так, чтобы максимизировать значение функции правдоподобия (функции, связывающей правдоподобие выборки, произведение вероятностей получения каждого из результатов выборки, со значениями параметров распределения).

**Линейная регрессионная модель**, помимо решения задачи регрессии, может также применяться для решения задач классификации. Для этого совершаются аналогичные действия, значения целевой переменной будут находиться от 0 до 1. При предсказании в случае получения значения выше 0.5, то ответ является положительным, иначе – отрицательным. Однако такой метод не будет эффективным по следующим причинам: прогнозируемое значение

является непрерывным, а не вероятностным, и линейная регрессия чувствительна к дисбалансу данных.

**Алгоритм случайного леса** основан на множестве деревьев решений и использует следующую стратегию:

- корневой узел каждого дерева содержит начальную выборку данных. Для каждого отдельного дерева эта выборка уникальна;
- на каждом узле выборка случайным образом разбивается на два подмножества для получения двух следующих узлов с меньшими выборками;
- каждое дерево растет до максимально возможного размера, пока на каждой ветви не будет исчерпана вся выборка;
- когда все деревья построены, осуществляется классификация объектов путем голосования. Каждое дерево «голосует» либо за то, что компонент содержит дефекты, либо за то, что дефектов нет. В итоге компонент признается дефектным либо недефектным в зависимости от того, за какой класс решения отдано больше голосов.

Алгоритм показывает высокую точность классификации на больших наборах данных.

**Наивный байесовский классификатор** – простой классификатор, основанный на применении теоремы Байеса с наивным предположением о независимости. Для каждого класса рассчитывается апостериорная вероятность по теореме Байеса (формула (3)), и класс с наивысшей апостериорной вероятностью будет являться результатом.

$$\mathrm{P}(c|x) = \frac{P(x|c)P(c)}{P(x)}, \tag{3}$$

где $P(c|x)$ – апостериорная вероятность данного класса;

$P(c)$ – априорная вероятность данного класса;

$P(x|c)$ – вероятность данного значения признака при данном классе;

*P(x)* – априорная вероятность данного значения признака.

То есть алгоритм также поддерживает многоклассовую классификацию. Несмотря на простоту, наивные байесовские классификаторы зачастую работают намного лучше нейронных сетей. К достоинствам также можно отнести малое количество данных, необходимых для обучения. Алгоритм обладает довольно высоким уровнем предсказания, производительность выше, чем у других простых алгоритмов, при этом требуется меньше обучающих данных [9].

Основная идея **метода опорных векторов** в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Выбирается такая гиперплоскость, которая разделяет классы наилучшим образом. Вектора, лежащие ближе всего, называются опорными векторами. То есть задачей является нахождение таких параметров, которые минимизируют расстояния до каждого класса. Метод не получил широкого применения в области определения дефектов ПО, так как является менее эффективным при прогнозировании программных ошибок [10].

Идея **градиентного бустинга** основана на идее преобразования моделей, которые обучились недостаточно хорошо, в модель с высоким показателем точности. Каждое новая модель обучается на модифицированных данных. Таким образом, точность увеличивается. Данная модель идентифицирует «слабые» модели на основе градиентных функций потерь [9].

С помощью **модели нечетной логики** можно формализовать величины, имеющие качественную основу, выявить причинно-следственные связи между регулируемыми параметрами и влияющими на них величинами и сформулировать прогноз в условиях неопределенности параметров прогнозирования [11]. Данная методология предполагает следующие шаги:

1. Определение входов и выходов создаваемой системы.

2. Определение функции принадлежности для каждой метрики.

3. Разработка нечетких правил и получение экспертного мнения для реализуемой нечеткой системы.

4. Агрегирование всех отдельных нечетких множеств для различных правил.

5. Поиск четкого значения путем дефаззификации агрегированного нечеткого множества.

Достоинством данной модели является то, что она использует как количественные, так и качественные показатели метрик для прогнозирования дефектов, недостатком − увеличение сложности вычислений в связи с увеличением входных показателей.

## 2.4 Метрики для оценки качества алгоритмов классификации

Для того, чтобы оценить качество алгоритмов, точность полученных значений, используются различные метрики [4]. Так как в данной работе рассматривается бинарная классификация, то классы можно интерпретировать как положительный и отрицательный при наличии и отсутствии дефектов соответственно. Рассмотрим некоторые из них для задачи классификации.

**Accuracy** – доля правильных ответов алгоритмов, рассчитывается по формуле (4) (отношение правильных прогнозов к их общему количеству). Это показатель, который описывает общую точность предсказания модели. Данная метрика плохо работает в задачах с неравными классами.

$$\text{a}ccuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (4)$$

где $TP$ – количество верно определенных объектов положительного класса;
$TN$ – количество верно определенных объектов отрицательного класса;
$FP$ – количество неверно определенных объектов положительного класса;
$FN$ – количество неверно определенных объектов отрицательного класса.

**Precision (точность)** – доля объектов, верно определенных моделью к положительному классу, вычисляется формуле (5). Эта метрика не позволяет присвоить всем объектам один и тот же класс.

$$precision = \frac{TP}{TP+FP}, \quad (5)$$

где $TP$ – количество верно определенных объектов положительного класса;
$FP$ – количество неверно определенных объектов положительного класса.

**Recall (полнота)** показывает, какую долю объектов положительного класса среди всех таких объектов нашла модель. Метрика демонстрирует способность алгоритма обнаруживать данный класс в целом. Вычисляется по формуле (6).

$$recall = \frac{TP}{TP+FN}, \quad (6)$$

где $TP$ – количество верно определенных объектов положительного класса;
$FN$ – количество неверно определенных объектов отрицательного класса.

**F-мера** представляет собой гармоническое среднее между точностью и полнотой – формула (7). То есть метрика объединяет в себе информацию и о точности, и о полноте, что дает более общую картину результата.

$$F = 2 \times \frac{precision \times recall}{precision + recall}, \quad (7)$$

## 2.5 Сравнение анализируемых алгоритмов

В результате анализа алгоритмов для предсказания количества дефектов ПО были выявлены преимущества и недостатки рассматриваемых алгоритмов,

их особенности. На основе этого составлена таблица 1, в которой представлены преимущества и недостатки данных алгоритмов.

Таблица 1 – Сравнительная таблица алгоритмов для предсказания количества дефектов ПО

| **Алгоритмы прогнозирования количества дефектов** | **Преимущества** | **Недостатки** |
|---|---|---|
| Линейная регрессия | Скорость; простота получения модели; легко интерпретируема | Требование равномерного распределения данных; данные связаны линейно |
| Модель роста надежности | Выполняется экстраполяция данных с учетом предположений о свойствах программы | Предполагается, что надежность ПО постоянно увеличивается; интенсивность проявления ошибок не изменяется |
| Метод исторических аналогий | Полезен, когда для системы трудно выявить конкретные правила | Скорость работы, точность результатов |
| Конструктивная модель качества | Модель позволяет делать более глубокий дефектный анализ | Субъективность экспертных оценок |

В результате проведенного сравнительного анализа выявлено, что наиболее эффективным является алгоритм линейной регрессии, который обеспечивает точность и высокую скорость работы алгоритмов модели.

Для сравнения алгоритмов классификации воспользуемся исследованиями из статьи [5], в которой представлены значения F-меры на различных наборах

данных. В таблице 2 представлены средние значения данной метрики для алгоритмов, также приведены их плюсы и минусы.

Таблица 2 – Сравнительная таблица алгоритмов для классификации дефектов ПО

| **Алгоритмы классификации дефектов** | **F-мера** | **Преимущества** | **Недостатки** |
|---|---|---|---|
| Логистическая регрессия | 80.11 | Скорость; простота получения модели; легко интерпретируема | Требование довольно больших размеров обучающей выборки |
| Линейная регрессия | — | | Плохо подходит для задачи классификации |
| Алгоритм случайного леса | 83.59 | Стабилен; более объективен; пропущенные значения не влияют на результат | Хороший результат в случае большой глубины деревьев; сложность вычислений |
| Наивный байесовский классификатор | 77.73 | Скорость; меньшее количество данных для обучения | Точность; предположение о независимости данных |
| Метод опорных векторов | 82.91 | Более «уверенная» классификация; хорошо изучен | Менее эффективен; неустойчивость к шуму |

| Алгоритмы классификации дефектов | F-мера | Преимущества | Недостатки |
|---|---|---|---|
| Классификатор градиентного бустинга | 85.81 | Точность; можно рассматривать различные функции потерь, любые базовые алгоритмы; простая оптимизация | Трудоемкость; без модификаций подстраивается под шум |
| Нечеткая логика | — | Использует как количественные, так и качественные показатели метрик | Увеличение сложности вычислений в связи с увеличением входных показателей |

Таким образом, для классификации дефектов ПО можно выделить алгоритм – классификатор градиентного бустинга, который в совокупности рассмотренных признаков является наиболее эффективным. Несмотря на высокую трудоемкость ввиду количества вычислений, он имеет достаточно преимуществ таких, как высокая точность, вариативность, простая оптимизация.

## 2.6 Выводы

Был выполнен обзор методов машинного обучения для определения дефектов ПО, приведены их описания, достоинства и недостатки. Составлена классификация с разделением методов на две группы, позволяющие решать задачи прогнозирования количества дефектов и определения наличия в программном компоненте дефекта. Для каждой из них выявлен наиболее эффективный алгоритм на основе сравнительного анализа.

Для решения задачи выявления количества дефектов наиболее целесообразно использовать алгоритм линейной регрессии, который является достаточно точным и производительным. Для классификации дефектов ПО наилучшим образом подходит классификатор градиентного бустинга, основными преимуществами которого являются высокая точность, возможность использования различных функций потерь и базовых алгоритмов, а также простая оптимизация.

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы была достигнута цель – определены наиболее эффективные алгоритмы машинного обучения для определения дефектов ПО, и решены поставленные задачи:

- изучена проблема обнаружения дефектов ПО;
- рассмотрен процесс прогнозирования дефектов ПО на основе машинного обучения;
- составлена классификация существующих методов;
- рассмотрены и проанализированы методы и модели машинного обучения на основе их классификации;
- приведена сравнительная таблица для рассмотренных методов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. International Software Testing Qualifications Board. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения. Версия 2.3 (от 9 июля 2014 года) [Текст] / ред. пер. Александр Александров. – 17 с.
2. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных [Текст] / П. Флах. – М.: ДМК Пресс. – 2015. – 402 с.
3. Юхименко Н.В. Обзор методов прогнозирования дефектов программного обеспечения [Текст] / Н.В. Юхименко, Ю.С. Белов // Программные продукты, системы и алгоритмы. – 2019. – №1 – С. 2.
4. Ebubeogu A.F. Predicting the number of defects in a new software version [Text] / A.F. Ebubeogu, P.L. Sai. – 2020.
5. Mitt S. Software Defects Prediction Using Machine Learning [Text] / S. Mitt, P. Nandit. – 2017.
6. Кирносенко С.И. Прогнозирование обнаружения дефектов в программном обеспечении [Текст] / С.И. Кирносенко, В.С. Лукьянов // Программные продукты и системы. – 2011. – № 3. – С. 68.
7. Sabnis P. Software Reliability Growth Model with Bug Cycle and Duplicate Detection Techniques [Text] / P. Sabnis, A. Kadam // Bharati Vidyapeeth Deemed Univ. College of Eng. – Pune, India, 2013. – PP. 345-349.
8. Dulal C.S. Software Defect Prediction Based on Classification Rule Mining [Text] / C.S. Dulal // Department of comp. sci. and Eng. National Institute of Technology Rourkela. – 2013. – PP. 19-65.
9. Mohammad A.I. Predicting Software Defects using Machine Learning [Text] / A.I. Mohammad // Techniques International Journal of Advanced Trends in Computer Science and Engineering. – 2020. – № 9(4). – PP. 6609-6616.
10. Junaid A.R. Predicting Software Defects Through SVM: An Empirical Approach [Text] / A.R. Junaid, S. Satwinder // IJSRD – International Journal for Scientific Research & Development. – 2017. – Vol. 5 – Issue 05 – PP. 1835-1838.

11. Шадрина В.В. Применение методов прогнозирования в технических системах [Текст] / В.В. Шадрина // Изв. ЮФУ: Технич. науки. – 2011. – № 2. – С. 141-145.